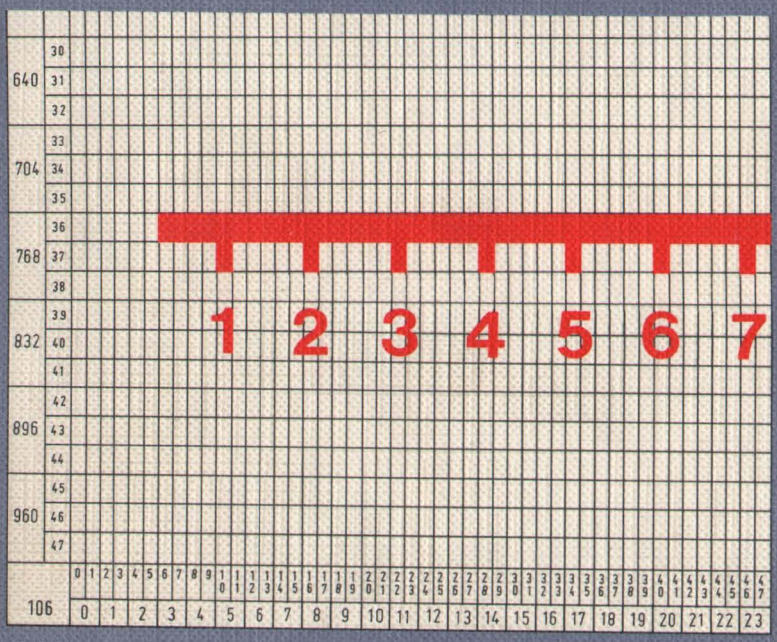
Franzis
Computer-Praxis

Busch
Basic für Aufsteiger

Der sichere Weg zum fortgeschrittenen Basic-Programm



Sie erweitern den Begriff „Befehl"
Die Programmierung
wird systematisiert
Aufgabenbeschreibung
Entwurf des Programmablaufes
Was ist DIN 66001?
Winkelfunktionen
Sie erweitern
das PRINT-Kommando
Die Behandlung von Strings
Sie verwandeln eine
Dezimalzahl in eine Binärzahl
Sie entwerfen ein Wörterbuch
Zeichnen mit PRINT
Sie zeichnen eine Figur
Sie durchsuchen den Bildschirm
Nützliche Programmier-Routinen
Programme aus Mathematik und
Geometrie
Programme
aus der Betriebswirtschaft
Matrizen-Operationen

Franzis'

Busch
Basic für Aufsteiger

In der Gruppe
FRANZIS Computer-Praxis
sind erschienen:

Busch, Basic für Einsteiger
Feichtinger, Mit Computern steuern
Haugg, Software-Engineering
Klein, Basic-Interpreter
Klein, Mikrocomputersysteme
Klein, Mikrocomputer Hard- und Softwarepraxis
Klein, Z-80 Applikationsbuch
Klein, Was ist Pascal?
Klein, Mikrocomputer selbstgebaut
Piotrowski, IEC-Bus
Plate, Pascal: Einführung – Programmentwicklung – Strukturen
Wunderlich, Erfolgreicher mit CBM arbeiten

Franzis Computer-Praxis

Rudolf Busch

Basic für Aufsteiger

Der sichere Weg zum fortgeschrittenen Basic-Programm

Mit 44 Abbildungen und 12 Tabellen

Franzis'

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Busch, Rudolf:
Basic für Aufsteiger: d. sichere Weg zum fortgeschrittenen Basic-Programm / Rudolf Busch. – München: Franzis, 1983.
(Franzis-Computer-Praxis)
ISBN 3-7723-7281-3

© 1983 Franzis-Verlag GmbH, München

Sämtliche Rechte – besonders das Übersetzungsrecht, an Text und Bildern vorbehalten.
Fotomechanische Vervielfältigungen nur mit Genehmigung des Verlages.
Jeder Nachdruck – auch auszugsweise – und jegliche Wiedergabe der Bilder sind verboten.

Druck: Franzis-Druck GmbH, Karlstraße 35, 8000 München 2
Printed in Germany · Imprimé en Allemagne.

ISBN 3-7723-7281-3

Inhalt

1 Vorwort ... 10

2 Sie erweitern den Begriff: „Befehl“ ... 11
2.1 Kommando ... 11
2.2 Statement ... 11
2.3 Funktionen ... 12
2.4 Operatoren ... 13

3 Die Programmierung wird systematisiert ... 14
3.1 Aufgabenbeschreibung ... 14
3.2 Entwurf des Programmablaufes ... 16
3.3 Zeichnen des Programmablaufes ... 17
3.4 Was ist DIN 66001? ... 20
3.5 Zusammenfassung ... 24

4 Sie lernen mehr über die mathematischen Fähigkeiten Ihres Computers ... 26
4.1 Die wissenschaftliche Schreibweise ... 26
4.2 Winkelfunktionen ... 28
4.3 Benutzerfunktionen ... 31
4.4 Spezielle Funktionen ... 32
4.4.1 Logarithmen ... 32
4.4.2 Die Exponentialfunktion ... 32
4.5 Vermischtes ... 35
4.6 Zusammenfassung ... 36

5 Neue Typen von Variablen ... 37

6 Sie erweitern das PRINT-Kommando ... 39
6.1 PRINT TAB (X) A ... 39
6.2 PRINT @ X,A PRINT AT X,A ... 39
6.3 PRINT USING...;A ... 40

7 Die Behandlung von Strings ... 43
7.1 ASC(...) ; CHR$(...) ... 45
7.2 LEN (...) ... 46
7.3 Stringausgaben ... 47
7.4 Codier- und Decodiermaschine ... 49
7.5 Welcher String steckt im String? ... 50
7.6 STRING$(N,...) ... 52
7.7 VAL(STRING) ... 52
7.8 STR$(...) ... 53
7.9 INKEY$, GET ... 54

8 Beispiele für String-Manipulationen ... 56
8.1 Ordnen von Zahlenkolonnen ... 56
8.2 Sie fügen das Datum in Ihr Programm ein ... 57
8.3 Sie verwandeln eine Dezimalzahl in eine Binärzahl ... 60
8.4 Sie verwandeln eine Binärzahl in eine Dezimalzahl ... 64
8.5 Sie schreiben Geldbeträge in Worten ... 66
8.6 Sie entwerfen ein Wörterbuch ... 73

9 Sie editieren ... 76
9.1 Was bedeutet EDIT? ... 76
9.2 Was beherrschen Sie schon? ... 76
9.3 Einige Beispiele ... 77

10 Der Computer als Zeichenmaschine ... 82
10.1 Zeichnen mit PRINT TAB... ... 82
10.2 Zeichnen mit PRINT ... 84
10.3 Zeichnen mit Strings ... 85
10.4 Eine neue Zeichenhilfe: PRINT@ ... 85
10.5 SET, RESET ... 87
10.6 Sie entwerfen ein Achsenkreuz ... 90
10.7 Sie entwerfen ein Balkendiagramm ... 94
10.8 Sie zeichnen eine mathematische Funktion ... 96
10.9 Sie zeichnen eine Figur ... 101
10.10 Die Figur bewegt sich! ... 103
10.11 Sie durchsuchen den Bildschirm ... 106
10.12 Sie arbeiten mit dem Zeichensatz Ihres Computers ... 108

11 Die Grenzen von BASIC ... 116

12 Wie spart man Speicherplatz? ... 118

13 Fehlersuche ... 122
13.1 Syntax-Fehler ... 122
13.2 Ablauf-Fehler ... 122
13.3 Logische Fehler ... 123

14 Programm-Bibliothek ... 126
14.1 Nützliche Programm-Routinen ... 126
14.1.1 Ausstieg aus den Schleifen ... 126
14.1.2 Warteschleifen ... 128
14.1.3 Aufhalten des Programms mit INPUT ... 128
14.1.4 Menütechnik mit ON...OPTO... ... 129
14.1.5 Sperren gegen Falscheingaben ... 130
14.1.6 Blockweise Ausgabe von Ergebnissen ... 131
14.1.7 Bildung von Summen, Durchschnittswerten, „N-Quadrat“ ... 132
14.1.8 Rundungen ... 133
14.1.9 Vereinzeln ... 133
14.1.10 Erzeugen von Zufallszahlen ... 135
14.1.11 Sie suchen aus einer Reihe von Zahlen die kleinste ... 135
14.1.12 Sie suchen aus einer Reihe von Zahlen die größte ... 136
14.1.13 Sie suchen aus einer Reihe von Zahlen die größte und kleinste ... 136
14.1.14 Allgemeines zu Sortierprogrammen ... 138

14.1.15 Sie sortieren Zahlen ... 138
14.1.16 Sie sortieren Strings ... 140
14.1.17 Das Sortieren wird komplizierter ... 141
14.1.18 Der Computer ermittelt automatisch die Anzahl der Daten ... 143
14.1.19 Sie lassen den Computer selbständig DIMensionieren ... 144
14.2 Programme aus Mathematik und Geometrie ... 145
14.2.1 Sie lösen Gleichungen mit zwei Unbekannten ... 145
14.2.2 Sie lösen Gleichungen mit drei Unbekannten ... 146
14.2.3 Was ist ein rechtwinkliges Dreieck? ... 148
14.2.4 Sie untersuchen die Gleichung: Y = A * X + B ... 150
14.2.5 Die Nutzanwendung: Sie ermitteln die Tendenz von Meßwerten ... 152
14.3 Programme aus der Betriebswirtschaft ... 157
14.3.1 Kapitalverzehr ... 157
14.3.2 Programm zur Ermittlung der Wertminderungs-Rate ... 158
14.3.3 Abschreibungsmethoden ... 160
14.3.4 Hochrechnungen ... 168
14.3.5 Währungstabellen und andere ... 172
14.4 Programme zur Datenverarbeitung ... 180
14.4.1 Sie „verwalten" ein Reitturnier – Umgang mit Listen ... 180
14.5 Matrizen-Operationen ... 214
14.5.1 Wir arbeiten mit Matrizen ... 217
14.5.2 Sie handeln mit Obst und Matrizen ... 221
14.5.3 Und nun sind Sie Malermeister ... 222

Literaturverzeichnis ... 227
Sachverzeichnis ... 228

SIEMENS
SIEMENS

Wichtiger Hinweis

Die in diesem Buch wiedergegebenen Schaltungen und Verfahren werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden*).
Alle Schaltungen und technischen Angaben in diesem Buch wurden vom Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. Der Verlag und der Autor sehen sich deshalb gezwungen, darauf hinzuweisen, daß sie weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen kann. Für die Mitteilung eventueller Fehler sind Verlag und Autor jederzeit dankbar.

---

*) Bei gewerblicher Nutzung ist vorher die Genehmigung des möglichen Lizenzinhabers einzuholen.

1 Vorwort

In diesem Band lernen Sie nun diejenigen BASIC-Befehle kennen, auf deren Behandlung ich im Band „BASIC für Einsteiger" aus gutem Grund verzichtet hatte.

Wiederum werden Sie anhand von zahlreichen Beispielen lernen, wie man mit diesen Befehlen umgeht und wie sie Ihnen helfen können, Probleme zu lösen.

Die Beispiele ebenso wie die fertigen, getesteten Programme sollen Ihnen keine Kochrezepte liefern, die Sie kommentarlos übernehmen. Vielmehr sollen Sie dazu angeregt werden, eigene Programmieraufgaben in Angriff zu nehmen. Sie werden dabei die Fähigkeit entwickeln, diese zielgerichtet zu analysieren und in ein lauffähiges BASIC-Programm umzusetzen. So wie Ihr Computer ein Werkzeug, ein Hilfsmittel ist, so ist auch BASIC ein Werkzeug, das Ihnen den Zugang zu Ihrem Computer erlaubt.

Ich habe irgendwo gelesen, daß in den knapp 25 Jahren, in denen höhere, problemorientierte Programmiersprachen existieren, drei „Weltsprachen" entstanden sind. Neben COBOL und FORTRAN – die überwiegend von professionellen Programmierern verwendet werden – zählt dazu BASIC. Wenn Sie dieses Buch durchgearbeitet haben, werden Sie sich also in guter Gesellschaft befinden.

Am Ende trennt Sie nur noch eines vom Perfektionisten: Übung, Übung und nochmal: Übung.

Viel Spaß mit BASIC!

Gifhorn, 1983 Rudolf Busch

2 Sie erweitern den Begriff: „Befehl“

Aus Gründen der Vereinfachung hatte ich im ersten Band alle BASIC-Worte schlicht einen „Befehl“ genannt, wenn Sie damit Ihren Computer zu einer ganz bestimmten Reaktion veranlassen wollten.

Es ist an der Zeit, diese „Befehle“ einmal genauer unter die Lupe zu nehmen; Ihnen die korrekte Bezeichnung für die verschiedenen Worttypen zu erläutern.

Sie lernen dadurch nicht unbedingt mehr oder besser zu programmieren. Aber wenn Sie vom BASIC-Fieber befallen werden, wollen Sie sicher mit Gleichgesinnten darüber reden oder die einschlägige Fachliteratur studieren. Und da sollten Sie schon die richtigen Begriffe beherrschen. Um zu verstehen – und um verstanden zu werden.

Das gesamte BASIC-Vokabular wird in vier Kategorien eingeteilt: in Kommandos, Statements, Funktionen und in Operatoren.

2.1 Kommando (englisch: Command)

Ein Kommando befiehlt dem Computer, mit Ihrer Eingabe unmittelbar etwas anzustellen. Und zwar auch ohne vorangestellte Zeilennummer. Aus Ihrer Erfahrung wissen Sie, daß z. B.:

RUN
LIST
PRINT
AUTO
DELETE
CONT

diese Wirkung auf die Arbeitsweise Ihres Computers haben. Diesen Typ von BASIC-Worten nennt man „Kommando“.

2.2 Statement

Hier ist bereits das englische Wort geschrieben und auf den Versuch einer Übersetzung verzichtet worden – weil ein Übersetzungsversuch eher verwirren würde.

Statements sind im Verlaufe von BASIC-Programmen untergebrachte Worte, bei deren Erkennen der Computer vereinbarte Entscheidungen fällt oder bestimmte Operationen ausführt.

IF...THEN... wäre demnach ein typisches Statement; aber auch
READ...DATA...
RESTORE
ON...GO TO... usw.

Leider gibt es – auch bei BASIC – selten eine Regel ohne Ausnahme(n).

Wie Sie an PRINT... nachvollziehen können, kann PRINT... sowohl als Kommando als auch als Statement eingesetzt werden. Weil Sie „schreiben" können: PRINT 3*5 (oder, abgekürzt: ? 3*5); sie verwenden dann PRINT als Kommando. Oder:

```
1Ø...
2Ø IF...
3Ø...
4Ø PRINT...;
```

hier verwenden Sie PRINT als Statement.

2.3 Funktionen (englisch: Function)

Worte dieser Art rufen in Ihrem Computer fest eingebaute Unterprogramme ab und erledigen diese.

Typische Funktionen sind z. B.
SQR(X)
SIN(X)
VAL(X$)
LEN(A$) usw.

Bei den oben aufgezählten Funktionen sind die Unterprogramme folgende:

SQR(X)	– Es wird die Quadratwurzel von X ausgerechnet.
SIN(X)	– Es wird der Sinus von X ausgerechnet.
VAL(X$)	– Es wird der Zahlenwert des String X$ ermittelt.
LEN(A$)	– Es wird ermittelt, aus wievielen Zeichen der String A$ besteht.

An diesen Beispielen erkennen Sie, daß Funktionen alle gleich aufgebaut sind:

– Sie bestehen aus einem Teil, der angibt, WAS zu tun ist.
  Z. B. SQR... = Quadratwurzel ziehen.
– Sie bestehen aus einem Teil, der aussagt, MIT WEM etwas zu tun ist.
  Diesen Teil nennt man das „Argument" der Funktion.
  Das *Argument muß immer in ( ) stehen!*

Bei SQR(X) ist also X das Argument der Funktion. Geben Sie ein: SQR X, so erfolgt wegen der fehlenden Klammer eine Fehlermeldung.

BASIC verfügt über eine enorme Anzahl von Funktionen, die sich sehr vielseitig einsetzen lassen. Sie werden darüber von Fall zu Fall noch mehr lernen; hier soll das Kapitel abgeschlossen werden mit ein paar Regeln, welche Formen die Argumente von Funktionen haben dürfen.

Erstes Beispiel: *Das Argument ist eine Variable*

```
10 A = 64
20 PRINT SQR(A)
```

Zweites Beispiel: *Das Argument ist eine Konstante (Zahl)*

```
...
50 PRINT SQR(64)
```

Drittes Beispiel: Das Argument ist ein arithmetischer Ausdruck. (Eine „Formel")

```
100 PRINT SQR(3↑2 + 4↑2)
```

Viertes Beispiel: *Das Argument ist eine (Standard-)Funktion*

```
10 A=16
20 PRINT LOG(SQR(A))
```

Beachten Sie hier die „Klammer-Regeln"!!

Das vierte Beispiel können Sie – solange Sie noch ungeübt sind, wie folgt schreiben:

```
10 A=16 : B=SQR(A)
20 PRINT LOG(B)
```

2.4 Operatoren (englisch: Operator)

Unter Operatoren versteht man in BASIC das, was Sie im Schriftdeutsch unter Satzzeichen wie Punkt und Komma verstehen.

Operatoren sind ebenso sorgfältig und überlegt einzusetzen, wie alles, was mit BASIC zu tun hat, wenn Sie von Ihrem Computer eine Rüge in Form einer Fehlermeldung vermeiden wollen.

PRINT TAB,(30);A führt zu einer Fehlermeldung, weil in diesem Statement Komma und Strichpunkt nichts zu suchen haben. Es muß heißen:
PRINT TAB(30)A.

Auch damit lernen Sie im weiteren Verlaufe dieses Buches noch umzugehen!

3 Die Programmierung wird systematisiert

Bereits nach wenigen Stunden Praxis mit BASIC erscheint einem das Programmieren so einfach zu sein, daß man Gefahr läuft, einen folgenschweren Fehler zu machen:

Man setzt sich vor sein Gerät und tippt drauflos.

Bei einfachen „Geradeaus"-Programmen mag das noch angehen. Aber mit jeder Schleife, mit jeder Verzweigung wächst die Gefahr, daß sich das Programm irgendwo festfährt, weil man als Programmierer nur eine begrenzte Zahl von „wenn" und „dann" im Kopf vorausdenken kann.

Ich gebe gerne zu, daß ich selbst am Anfang ebenso vorgegangen bin. Erst wenn's mal nicht weiterging, habe ich zu Papier und Bleistift gegriffen und versucht, den Ablauf des Programms nachzuvollziehen, um logische Fehler aufzuspüren. Oder ich habe ganz einfach versucht, herauszufinden, was der Computer eigentlich mit dem eingegebenen Programm gemacht hat.

Wenn Sie ebenso beginnen, schadet das zunächst nicht viel – schließlich verschwenden Sie Ihre eigene Zeit. Aber je weiter Sie fortschreiten, um so komplexere Programme werden Sie in Angriff nehmen. Früher oder später kehren Sie dann auf den Pfad der Tugend zurück.

Diesen will ich Ihnen beispielhaft vorzeichnen.

3.1 Aufgabenbeschreibung

Beginnen Sie damit, das Problem, zu dessen Lösung Sie ein Programm entwerfen wollen, in Worten und Sätzen zu beschreiben. Durchdenken Sie die Aufgabenstellung solange, bis Sie genau verstanden haben, was Sie wollen und was Sie von Ihrem Computer erwarten.

Sie werden bemerken, daß ich das bei fast allen Programmen so mache.

Dabei ist es nützlich, ein paar Sätze, Bedingungen, Formeln etc. aufs Papier zu bringen.

Ich zeige Ihnen hier *Tabelle 1:*

Tabelle 1:

...so sieht die zu ordnende Tabelle zu Beginn aus...

I	A (I)	B (I)
1	Ø	9
2	Ø	8
3	Ø	5
4	Ø	1
5	2	9
6	2	7
7	3	12
8	3	1
9	3	19
1Ø	3	8

Sie finden darin N (im Beispiel ist N = 10) Wertepaare A(I) und B(I), die jeweils so zusammengehören wie Vorname und Familienname.

Sie sehen, daß die Liste der A(I) bereits aufsteigend geordnet ist, während die der B(I) keine Ordnung erkennen läßt. Sie sollen nun ein Programm entwerfen, das am Ende die Liste wie folgt umschreibt (*Tabelle 2*):

Tabelle 2

...und so soll Ihr Computer die Liste umstellen.

I	A (I)	B (I)
1	Ø	1
2	Ø	5
3	Ø	8
4	Ø	9
5	2	7
6	2	9
7	3	1
8	3	8
9	3	12
1Ø	3	19

Wieder im Klartext:

Die Elemente A(I),B(I) sollen so umgestellt werden, daß auch die B(I) innerhalb jeder Gruppe gleicher A(I) in aufsteigender Reihenfolge geordnet sind.

Gefordert ist ferner, daß das Programm dazu für beliebig viele Wertepaare – also nicht nur für die zehn des Beispiels!! – gültig ist.

Was ist zu tun?

- Sie haben drei Blocks untereinander gleicher A(I), nämlich Ø (4mal), 2 (2mal) und 3 (4mal).
- Für jeden dieser Blocks müssen Sie getrennt und nacheinander die B(I) aufsteigend umstellen.
- „Aufsteigend umstellen" heißt aber nichts anderes als: SORTIEREN.
- Um eine Reihe von Zahlen oder Begriffen zu sortieren, müssen Sie wissen:

– WO ist jeweils der ANFANG des Sortierprogramms?
– WIEVIELE Elemente sind zu sortieren?
– WO ist jeweils das ENDE des Sortiervorgangs?

Für unser Übungsprogramm lautet die Antwort auf diese drei Fragen:

zu sortierende Gruppe:	Beginn bei I =	Anzahl der Elemente	Ende bei I =
A(I) = Ø	1	4	4
A(I) = 2	5	2	6
A(I) = 3	7	4	1Ø

Wir wollen hier – in diesem Kapitel – zunächst ein Programm entwerfen, das die obigen Fragen beantwortet. Den eigentlichen Sortiervorgang heben wir uns für ein späteres Programm auf.

3.2 Entwurf des Programmablaufes

Ganz grob muß das Programm wie folgt ablaufen:

- Ist A(1) gleich A(2)?
- Wenn ja:
  Ist A(2) gleich A(3)?
- Diese Fragerei setzen Sie fort, bis Sie erkennen: A(4) ist NICHT gleich A(5).
- Sie merken sich den Punkt (I !!), bei dem die NEIN-Aussage erfolgte...
- ...und haben damit das ENDE der gleichen A(I) gefunden.
- Sie haben festgehalten, WO Sie ANGEFANGEN haben...
- Sie haben mitgezählt, WIE OFT A(I) GLEICH A(I+1) war.

Damit können Sie bereits die drei gesuchten Antworten für den ersten Block gleicher A(I) hinschreiben.

Und nach dem gleichen Schema den zweiten Block gleicher A(I) durchmustern usw.

3.3 Zeichnen des Programmablaufes

Bei Beginn einer Programmentwicklung müssen wir damit rechnen, daß sich Fehler einschleichen und wir das Programm noch (oft?) ändern müssen.

Es genügt deshalb zunächst ein grober Entwurf, den wir erst ganz zum Schluß – nach erfolgreichem Probelauf – in seine endgültige Form bringen.

Hier ist unser Entwurf, *Abb. 1:*

Hier beginnen wir mit 4 Zählschleifen.
I zählt fortlaufend alle geprüften Elemente.
P zählt die Anzahl gleicher A(I)
S ist ein „Hilfszähler", er soll später den Anfang ermitteln.
X ist der Zähler zur Ermittlung des Endes.
Alle Zähler stehen zu Beginn auf Ø

Hier erfolgt die Prüfung auf gleiche A(I). Da wir mit I=Ø beginnen, ist A(1) gleich A(I + 1) usw.!

Nach jeder „Entscheidung" werden die Zähler um 1 erhöht...

...und es wird weiter gesucht!

Auch nach einer NEIN-Entscheidung werden die Zähler um 1 erhöht. Beachten Sie die Erhöhung von X um P!!
X war zu Anfang Ø und muß am Ende eines Blocks gleicher A(I) um deren Anzahl P größer sein.
Die Ergebnisse des ersten Blocks werden hingeschrieben...

...und der nächste Block A(I) wird durchsucht!

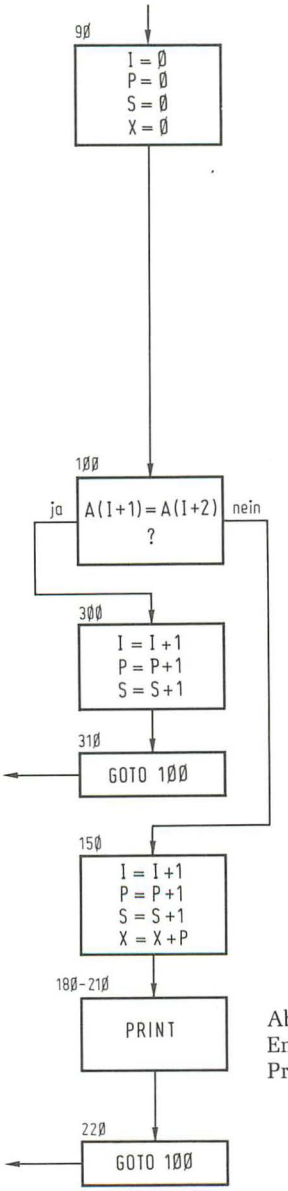


Abb. 1
Entwurf des Programmablaufes

Es erweist sich beim Programmieren als überaus nützlich, ein Programm in möglichst kleinen Blöcken zu entwickeln und diese vor der Fortsetzung der Arbeit vor allem auf Ihre Lauffähigkeit zu testen.

Damit Sie das hier tun können, müssen Sie zunächst die ursprünglichen Daten aus *Tabelle 1* in Ihren Computer eingeben.

Dazu schlage ich Ihnen folgendes Programm vor:

```
5 CLS
10 DIM A(20):DIM B(20)
20 N=10
30 FOR I = 1 TO N
40 INPUT "A(I) =";A(I)
50 INPUT "B(I) =";B(I)
60 NEXT I
```

Lassen Sie dieses Programm laufen und geben Sie die Daten aus Tabelle 1 für A(I) und B(I) ein!

Setzen Sie nun Ihr Programm fort mit den Programmzeilen 90, 100, 300, 310, 150 und 220 aus Abbildung 1!

Für den „PRINT-Block" – Zeilen 180 bis 210 – schlage ich vor:

```
180 PRINT"ENDE DER GLEICHEN A(I) BEI I =";I
190 PRINT"ANZAHL DER ZU SORTIERENDEN ELEMENTE P =";P
200 PRINT"BEGINN DES SORTIERENS BEI S - P + 1 =";S - P + 1
210 PRINT"ENDE DES SORTIERENS BEI X = X + P GLEICH :"; X
```

Gewiß waren Sie versucht, in Zeile 200 den Beginn des Sortierens auf I = 1 zu legen? Das wäre sogar richtig gewesen. Allerdings nur für den ersten Block gleicher A(I)! Sie sehen gleich klarer!

Wenn Sie dieses Programm starten, werden Sie über das Ergebnis den Kopf schütteln! Der Computer wird ausgeben:

```
ENDE DES SORTIERENS BEI X = X + P GLEICH 20
? BS ERROR IN 100
```

Und damit haben Sie den ersten Gedankenfehler entdeckt.
Das Programm sucht bis I = 20, weil in Zeile 10 DIMensioniert ist: A(20), B(20)
Wir haben dem Computer zu sagen vergessen, wann er mit der Suche aufhören soll!

Er soll offensichtlich mit der Suche aufhören, wenn I = N ist.
Fügen Sie in Ihr Programm ein:

```
305 IF I = N THEN 1000
215 IF I = N THEN 1000
1000 END
```

Damit ändert sich unser Entwurf zum Ablaufplan wie folgt (*Abb. 2*).

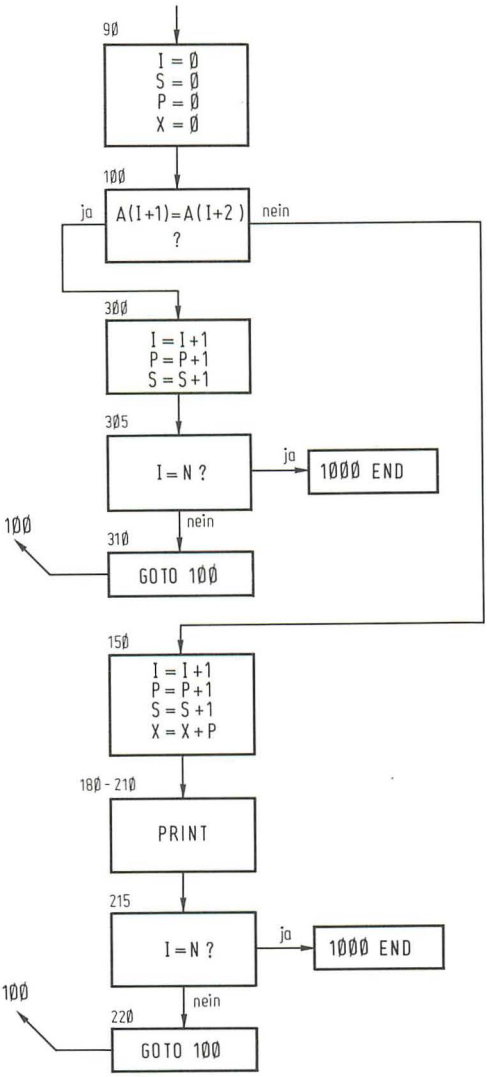


Abb. 2 Der Ablaufplan wird korrigiert

Ein erneuter Probelauf bringt ans Licht, daß nur die Antworten für den ersten Block gleicher A(I) richtig sind.

Der Fehler ist rasch gefunden:
Wenn P jeweils die gleichen A(I) zählt, dann müssen wir den Zähler wieder auf Ø stellen, sobald ein Block gleicher A(I) verlassen bzw. bevor ein neuer begonnen wird.

Ergänzen Sie Ihr Programm:

```
212 P = Ø
```

Und nun können Sie mit dem Ergebnis wohl zufrieden sein! Prüfen Sie die „Allgemeingültigkeit" des Programms, indem Sie die Blocks gleicher A(I) verändern oder einmal mehr als 1Ø Wertepaare eingeben! (dann aber Zeile 2Ø ändern!)

Wenn Sie sich die in diesem Abschnitt beschriebene Vorgehensweise zu eigen machen, haben Sie für Ihre weitere Arbeit viel gelernt.

Wenn Sie auch den nächsten durcharbeiten, werden Ihre Ablaufpläne sogar von anderen verstanden – und Sie verstehen die von Ihren Kollegen!

3.4 Was ist DIN 66001?

DIN... steht für *D*eutsche *I*ndustrie*N*orm. Und in DIN 66001 sind Sinnbilder für Programmablaufpläne vorgeschlagen, die diesen eine größere Transparenz verleihen. Hatten wir immer nur simple Rechtecke verwendet, so ist hier die Anzahl der Symbole etwas vergrößert worden (*Abb. 3*).

Unser Übungsprogramm gewinnt durch die Verwendung der genormten Symbole ein professionelles Aussehen (*Abb. 4*).

Oder gefällt's Ihnen besser so (*Abb. 5*)?

● Programmdokumentation

Haben Sie Ihr Programm erfolgreich getestet, so sollten Sie es schriftlich festhalten; Sie sollten es „dokumentieren".

Das ist aus mancherlei Gründen zweckmäßig:
- Das Programm kann auch von anderen Leuten verstanden werden.
- Sie selbst können es auch nach längerer Pause wieder verstehen – was garnicht so selbstverständlich ist, wie es scheint!

Zu der Dokumentation zählt der soeben behandelte

Sinnbild	Bedeutung	Beispiele
	Flußlinie verbindet die Sinnbilder untereinander. Der Pfeil zeigt in Richtung des Programmablaufs.	TEXT TEXT X
1 A	Übergangsstelle wird ein Programm an anderer Stelle, z.B. auf einem anderen Blatt, fortgesetzt, so weisen gleich beschriftete Kreise auf die Übergangsstelle hin.	1 von... nach... 1
TEXT	Allgemeine Operation	A=1 B=Ø GO TO...
TEXT	Verzweigung der Text enthält eine Bedingung, die nur mit JA oder NEIN zu beantworten ist	ja A = Ø ? nein
TEXT	Grenzstelle	START END
TEXT	Eingabe	INPUT A
TEXT	Ausgabe (insbesondere Drucker)	PRINT A*B
Zeilen-Nr.	Zeilen-Nummer kann im Flußplan entfallen oder wie nebenstehend angebracht werden	1ØØ A = 17 1ØØ A = 17 1ØØ END

Anmerkung :
zur Erleichterung der Zeichenarbeit gibt es passende Schablonen im Bürofachhandel, z.B. STANDOGRAPH Nr. 391

Abb. 3 Symbole für Ablaufpläne nach DIN 66001

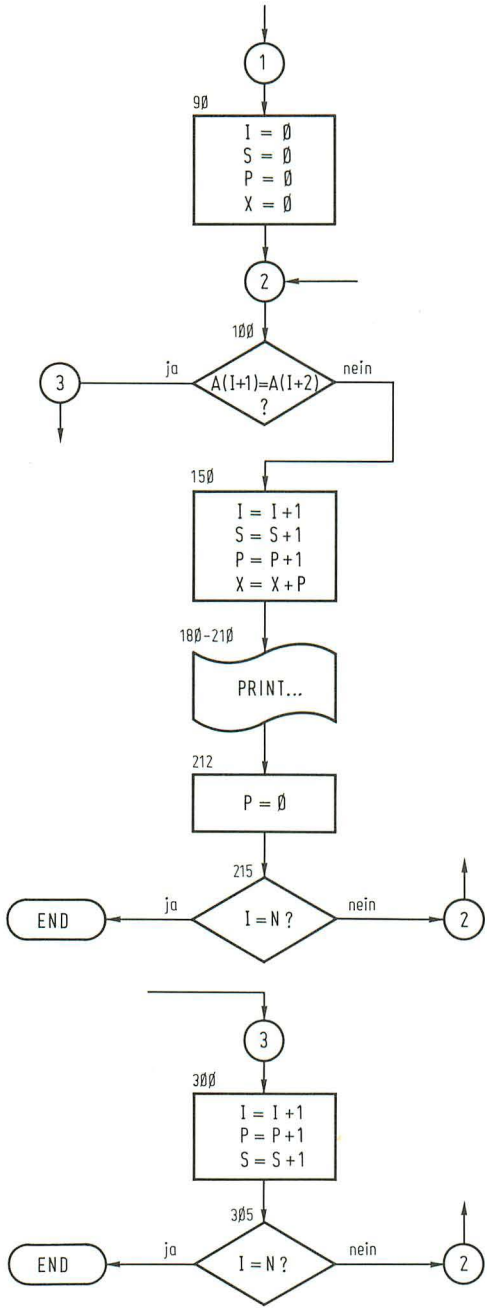


Abb. 4 Flußplan des Übungsprogramms mit Symbolen nach DIN 66001

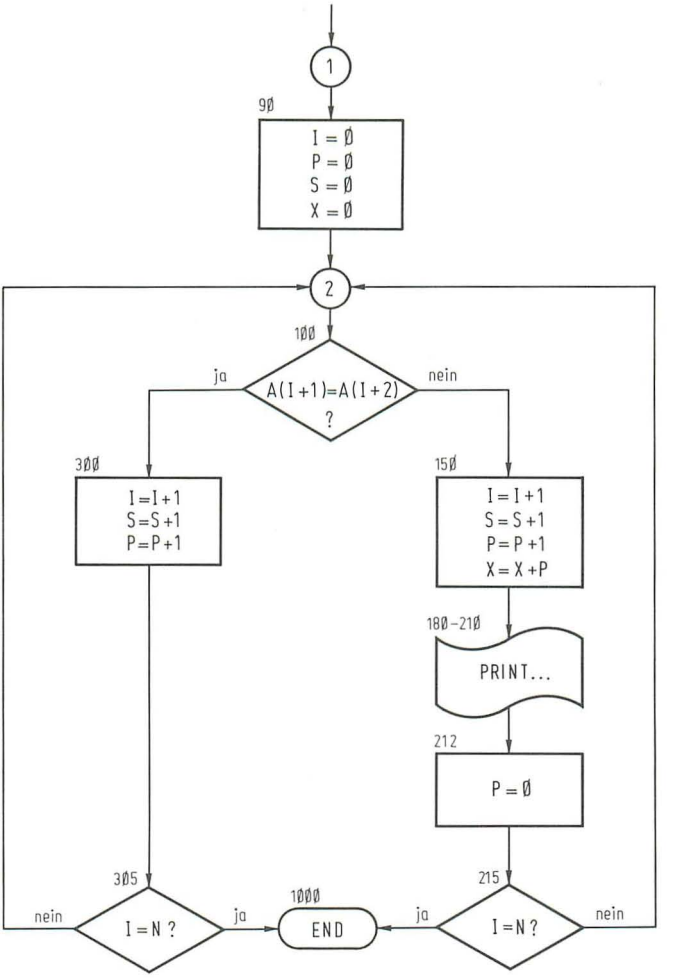


Abb. 5 Andere Darstellungsart der Abb. 4

● **Ablaufplan**

Ferner ein zeilenweises

● **Programm-Listing** (Liste der Computereingabe bei der Programmierung). Dieses fasse ich am Ende dieses Abschnittes noch einmal zusammen.

Zur Dokumentation zählt ferner ein

● **Kommentar,** in dem die Aufgabe beschrieben ist. In dem z. B. verwendete Formeln in ihrer mathematischen Schreibweise aufgeführt und „computerge-

recht" umgeschrieben sind. Und in dem ausgewählte Variablen in Ihrer Bedeutung (wer zählt was?) beschrieben sind usw.

Das kann in einem eigenen Text geschehen oder mit Hilfe von REM innerhalb des Programms.

Falls Sie über einen Drucker verfügen, können Sie schließlich Ihre Programmdokumentation mit einem Ausdruck des Programms sowie mit einem Probelauf abrunden!

Und nun das versprochene Programm-Listing:

```
5 CLS
10 DIM A(20): DIM B(20)
20 N = 10
30 FOR I = 1 TO N
40 INPUT"A(I) = ";A(I)
50 INPUT"B(I) = ";B(I)
60 NEXT I
90 I = 0 : P = 0 : S = 0 : X = 0
100 IF A(I + 1) = A (I + 2) THEN 300
150 I = I + 1 : S = S + 1 : P = P + 1 : X = X + P
180 PRINT"ENDE DER GLEICHEN A(I) BEI I =";I
190 PRINT"ANZAHL DER ZU SORTIERENDEN ELEMENTE P =";P
200 PRINT"BEGINN DES SORTIERENS BEI S - P + 1 =";S - P + 1
210 PRINT"ENDE DES SORTIERENS BEI X=X+P GLEICH";X
212 P = 0
215 IF I = N THEN 1000
220 GO TO 100
300 I = I + 1 : S = S + 1 : P = P + 1
305 IF I = N THEN 1000
310 GO TO 100
1000 END
```

3.5 Zusammenfassung

Früher oder später werden Sie auf Ihrem Weg zum Profi also Ihre Programme wie folgt aufbauen:

- Problemanalyse und -beschreibung.
- Entwurf des Ablaufplans.

- Entwurf des Programms (Listing, Computereingabe).
- Testen des Programms bei jedem sinnvoll erscheinenden Teilstück. Bei Fehlern korrigieren!
- Dokumentation des lauffähigen Programms.
- Ablage in Ihrer Programmbibliothek.

4 Sie lernen mehr über die mathematischen Fähigkeiten Ihres Computers

4.1 Die wissenschaftliche Schreibweise

Entlocken Sie Ihrem Computer einmal eine Antwort auf folgende Eingabe:

? 1ØØØØØØ * 15Ø

Die Antwort wird lauten:
1.5E+Ø8 oder, bei anderen Computermodellen 1.5E8
Erwartet hatten Sie wohl 15ØØØØØØØ ?

Finden Sie zunächst heraus, WARUM der Computer eine noch unverständliche Antwort gibt!

Auf die Eingabe ? 1ØØØØØ * 9 antwortet der Computer mit 9ØØØØØ
? 1ØØØØØ * 9.99999 999999
? 1ØØØØØ * 1Ø 1E+Ø6 –
und nicht 1ØØØØØØ!

999999 ist eine Zahl mit sechs Stellen, 1ØØØØØØ ist eine Zahl mit sieben Stellen. Der Computer wechselt also die Darstellungsart, sobald die Ausgabe mehr als sechs Stellen „lang" ist.

Sie können statt 1ØØ schreiben : $1Ø^2$
statt 1ØØØØØØ : $1Ø^6$

Der Computer kann das nicht.

Da der Computer ab sechs Stellen automatisch in die Potenzschreibweise – in die exponentielle Schreibweise – übergeht, macht er Sie durch das Zeichen E darauf aufmerksam.

1ØØØØØØ wird also statt $1Ø^6$ zu 1 E + Ø6

Solange Sie noch ungeübt in dieser Darstellungsart sind, merken Sie sich das folgende Schema:

1ØØØØØØ

hat 6 Nullen, schreibt man in der Mathematik auch als $1Ø^{6}$

schreibt der Computer als 1 E+Ø6

1ØØØØØØØ

hat 8 Nullen, schreibt man in der Mathematik auch als $1Ø^{8}$

schreibt der Computer als 1 E+Ø8

Daß man in dieser Darstellungsart auch „Rechnen" kann, erkennen Sie nach Abarbeiten der folgenden Aufgaben, aus denen gleichzeitig die Rechenregeln zu erkennen sind.

Ihre Eingabe:	Das gibt der Computer aus:
? 1.5E+Ø6 + 1.5E+Ø6	3E+Ø6
? 1.5E+Ø6 − .5E+Ø6	1E+Ø6
? 1.5E+Ø6 * 2E+Ø6	3E+12
? 1.5E+Ø6 / 1E+Ø3	1.5E+Ø3 (bzw. 15ØØ)
? 1.5E+Ø6 / 1E+Ø9	1.5E−Ø3

Wenn Sie das Ergebnis im letzten Beispiel 1.5E−Ø3 – betrachten, wissen Sie wohl, warum der Computer ein Plus oder Minus vor den Exponenten setzt?

$1.5*1Ø^{6} / 1.Ø * 1Ø^{9}$ ist: $\frac{15ØØØØØ}{1ØØØØØØØØØ}$ oder $\frac{1.5}{1ØØØ}$ oder Ø.ØØ15

Das Vorzeichen für den Exponenten wechselt also von Plus nach Minus, sobald die Zahl *kleiner als Null* wird.

Merke:

- Bei positiven Exponenten kann bei der Eingabe das Plus-Zeichen entfallen.
- Bei negativen Exponenten *muß* das Minus-Zeichen eingegeben werden.
- Die führende Null im Exponenten kann bei der Eingabe entfallen!
- Beispiel: ?1.5E6/1E9 ergibt 1.5E−Ø3

Schließen wir für den Umgang mit der Exponentialschreibweise gleich noch zwei Merkregeln an:

- Sie schreiben $1\emptyset^3$ statt $1*1\emptyset^3$
  Für Ihren Computer muß jedoch vor E mindestens eine Zahl stehen! Also: für $1\emptyset^3$ ist E3 verboten; es muß heißen 1E3 oder 1E+Ø3.
- Für den Exponenten sind nur ganze Zahlen zwischen +37 und −37 erlaubt. Dezimalzahlen und Brüche sind als Exponenten verboten. Also: 1E.7 oder 1E1/3 sind verboten!

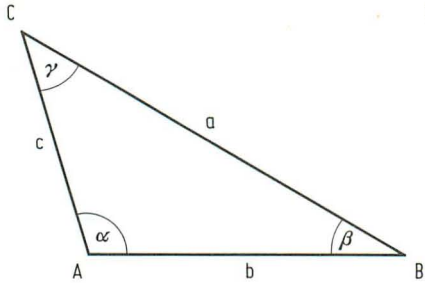
4.2 Winkelfunktionen

Die folgende Abbildung habe ich in meinem alten „Mathebuch" im Kapitel „Winkelfunktionen" gefunden (*Abb. 6*).

Es geht darin um die Berechnung von Winkeln und Seiten im schiefwinkligen Dreieck; die Formeln unter der Zeichnung geben dabei die gegenseitigen Abhängigkeiten an.

„sin.." und „cos..." sind dabei Abkürzungen für SINUS und COSINUS; „Werte", die man üblicherweise Tabellenbüchern entnehmen kann.

Falls Sie sich für weitere Details interessieren, zeigt Ihnen *Abb. 7* die Zusammenhänge etwas ausführlicher.



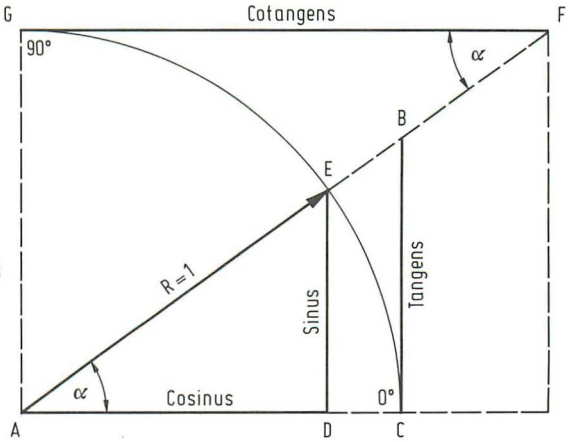
Sinus - Satz :

$$\frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma}$$

Cosinus - Satz :

$$a^2 = b^2 + c^2 - 2bc \cdot \cos\alpha$$
$$b^2 = a^2 + c^2 - 2ac \cdot \cos\beta$$
$$c^2 = a^2 + b^2 - 2ab \cdot \cos\gamma$$

Abb. 6 Beziehungen zwischen Winkeln und Seiten im schiefwinkligen Dreieck



Winkelfunktionen im 1. Quadranten des Einheitskreises (R = 1)

$\sin\alpha$ = DE
$\cos\alpha$ = AD
$\tan\alpha$ = BC
$\cot\alpha$ = FG

Abb. 7 Winkelfunktionen im Einheitskreis

Wollen Sie den SINUS eines Winkels wissen, brauchen Sie Ihren Computer nur zu fragen:

? SIN(A) – wobei A für den Winkel steht, dessen SINUS Sie wissen wollen.

Aber Achtung!
Während Sie gewohnt sind, Winkel in GRAD zu betrachten, erwartet der Computer Ihre Eingabe für A im BOGENMASS!

Zwischen Winkel (in Grad) und Winkel (im Bogenmaß) besteht folgender Zusammenhang:

- Bogenmaß = π/18Ø * Grad
- Grad = 18Ø/π * Bogenmaß
- π/18Ø = .Ø174533
- 18Ø/π = 57.2958

Mit dem folgenden Programm können Sie sich das Bogenmaß für einige „gängige" Winkel einmal ausrechnen lassen.

```
1Ø PI = 3.14159265 : F = PI/18Ø
2Ø PRINT"GRAD", "BOGENMASS", "GRAD", "BOGENMASS"
3Ø PRINT"=========================================
====="
4Ø FOR I = 15 TO 36Ø STEP 15
5Ø PRINT I, I*F,
6Ø NEXT
```

Was hat nun Ihr Computer zum Thema Winkelfunktionen noch zu bieten? „Fest eingebaut" sind die Funktionen SINUS, COSINUS und TANGENS; bei manchen Modellen auch noch ARCUSTANGENS.

Sie fordern diese Funktionen wie folgt von Ihrem Computer an:

SIN(A)
COS(A)
TAN(A)
ATN(A)

Dabei setzen Sie den Winkel A jeweils im Bogenmaß ein!
Und wie steht es etwa um den COTANGENS aus Abb. 7?
Nun, den kann Ihr Computer ausrechnen nach der Formel:
COTANGENS = 1/TANGENS.

Ebenso, wie Sie hier aus dem TANGENS die COTANGENS-Funktion errechnen lassen, können Sie aus den drei (bzw. vier) fest eingebauten Funktionen die verrücktesten Sachen ausrechnen lassen.

Falls Sie dafür Bedarf haben, können Sie sich der *Tabelle 3* bedienen; in dieser sind die Beziehungen in der rechten Spalte schon „computergerecht" für Ihre Eingabe umgeschrieben.

Tabelle 3: Spezielle Winkelfunktionen

Mathematische Funktion	Funktion	dargestellt mit BASIC
SEKANS	SEC(X)	= 1/COS(X)
KOSEKANS	CSC(X)	= 1/SIN(X)
KOTANGENS	COT(X)	= 1/TAN(X)
ARKUSSINUS	ARCSIN(X)	= ATN(X/SQR(−X*X+1))
ARKUSKOSINUS	ARCCOS(X)	=−ATN(X/SQR(−X*X+1))+1.57Ø8
ARKUSSEKANS	ARCSEC(X)	= ATN(SQR(X*X−1))+(SGN(X)−1)* 1.57Ø8
ARKUSKOSEKANS	ARCCSC(X)	= ATN(1/SQR(X*X−1))+(SGN(X)−1)* 1.57Ø8
ARKUSKOTANGENS	ARCCOT(X)	=−ATN(X)+1.57Ø8
HYPERBELSINUS	SINH(X)	= (EXP(X)−EXP(−X))/2
HYPERBELKOSINUS	COSH(X)	= (EXP(X)+EXP(−X))/2
HYPERBELTANGENS	TANH(X)	=−EXP(−X)/(EXP(X)+EXP(−X))*2+1
HYPERBELSEKANS	SECH(X)	= 2/(EXP(X)+EXP(−X))
HYPERBELKOSEKANS	CSCH(X)	= 2/(EXP(X)−EXP(−X))
HYPERBELKOTANGENS	COTH(X)	= EXP(−X)/(EXP(X)−EXP(−X))*2+1
INVERSER HYPERBEL-SINUS	ARCSINH(X)	= LOG(X+SQR(X*X+1))
INVERSER HYPERBEL-KOSINUS	ARCCOSH(X)	= LOG(X+SQR(X*X−1))
INVERSER HYPERBEL-SEKANS	ARCSECH(X)	= LOG((SQR(−X*X+1)/X))
INVERSER HYPERBEL-KOSEKANS	ARCCSCH(X)	=LOG((SGN(X)*SQR(X*X+1)+1)/X)
INVERSER HYPERBEL-KOTANGENS	ARCCOTH(X)	= LOG((X+1)/X−1))/2
INVERSER HYPERBEL-TANGENS	ARCTANH(X)	= LOG((1+X)/(1−X))/2

4.3 Benutzerfunktionen

Ihr Computer[1]) erlaubt es Ihnen, eigene Funktionen, sogenannte „Benutzerfunktionen" zu definieren und in Ihr Programm zu übernehmen. Das ist besonders dann von Nutzen, wenn eine „lange" Formel im Verlaufe eines Programms öfter benötigt wird.

Für dieses „Selbst-Definieren" dient in BASIC der Ausdruck:
DEF...

Dabei ist der Name der Funktion FN, gefolgt von einem Variablennamen. Nehmen Sie an, in einem Ihrer Programme kommt die Beziehung

A↑ 2 + B↑ 2 sowie
SQR(A ↑ 2 + B ↑ 2) öfter vor, dann können Sie daraus eine Selbst-DEFinierte Funktion machen.

Nutzen Sie diese Möglichkeit, dann müssen Sie die DEFinierte Funktion an den Anfang des Programms stellen bzw. im Programm unterbringen, *bevor* Sie diese benutzen.

```
100 INPUT"A=";A
110 INPUT"B=";B
```

```
120 DEF FNA(S) = A ↑ 2 + B ↑ 2 : S = FNA(S)
130 DEF FNW(P) = SQR(A ↑ 2 + B ↑ 2) : P = FNW(P)
```

Nun können Sie nicht nur aufrufen:

```
150 PRINT"S =";S
160 PRINT"P =";P
```

Sie können auch mit S und P weiterrechnen. Beispielsweise:

S + P oder S/P oder S*P ↑ 2 usw., immer werden für S bzw. für P die von Ihnen oben DEFinierten Funktionen eingesetzt.

Versuchen Sie's einmal! Verwenden Sie für A und B Werte, die Sie noch „im Kopf" verfolgen können!

*) DEF FN... ist NICHT bei jedem Computer verfügbar!

4.4 Spezielle Funktionen

Es geht immer noch um die mathematischen Fähigkeiten Ihres Computers. Sehen wir uns an, was noch in ihn hineingebaut wurde!

4.4.1 Logarithmen

Ein Logarithmus ist in der Mathematik ein Potenzexponent. Für diesen grauslichen Ausdruck gilt die allgemeine Beziehung:

$a^m = c$
(In Worten: a hoch m gleich c)

Setzen Sie a zu 1Ø
m zu 2
dann wird c zu 1ØØ; $1Ø^2 = 1ØØ$.

Es gilt ferner:
$a_{\log c = m}$
(Das lesen Sie: Logarithmus c zur Basis a ist m)

Als Basis für einen Logarithmus kann jede Zahl genommen werden. Dabei bilden alle Logarithmen gleicher Basis ein Logarithmensystem.

Ist diese Basis e = 2.71828..., so spricht man von „natürlichen Logarithmen", abgekürzt: ln

(e ist entwickelt aus: 1 + 1/1 + 1/1*2 + 1/1*2*3 + 1/1*2*3*4...)

Die natürlichen Logarithmen werden hauptsächlich in der „höheren" Mathematik eingesetzt; z. B. in statistischen Berechnungen.

Der langen Rede kurzer Sinn: Ihr Computer besitzt die Fähigkeit, zu von Ihnen eingegebenen Argumenten den natürlichen Logarithmus zu berechnen.

Sie rufen diese Funktion ab mit:

LOG(A). Fragen Sie also: ?LOG(1Ø), so antwortet Ihr Computer mit 2.3Ø259; der natürliche Logarithmus von 1Ø ist 2.3Ø259

Bevor Sie lernen, was Sie mit dieser Fähigkeit anfangen können, lesen Sie noch den nächsten Abschnitt!

4.4.2 Die Exponentialfunktion

Geben Sie einmal ein:

```
1Ø E = 2.71828
2Ø PRINT" E HOCH ZWEI IST :";E ↑ 2
3Ø PRINT"E HOCH ZWEI IST ABER AUCH :"; EXP(2)
```

In beiden Fällen ist das Ergebnis 7.389Ø6

Mit EXP.. haben Sie die Exponentialfunktion e^X kennengelernt. Die Größe e ist dabei die gleiche, die vorhin beschrieben wurde.

e^2 = EXP(2) = 7.389Ø6
e^6 = EXP(6) = 4Ø3.429 usw.

Und nun zur Nutzanwendung von LOG.. und EXP...!

Als es noch keine (Taschen-)Rechner gab und selbst Rechenschieber seltene Instrumente waren, bediente man sich zur Lösung etwa der Aufgabe: $\sqrt[3]{25}$ = ? der „Logarithmentafel" und löste dieses Rechenproblem in folgenden Schritten:

1. Aus der Logarithmentafel wird für die *Zahl 25* der Logarithmus ermittelt.
2. Dieser wird *durch 3 geteilt.*
3. Man erhält dadurch den *Logarithmus* des gesuchten *Ergebnisses.*
4. Zu diesem Logarithmus sucht man in der gleichen Tafel die zugehörende Zahl. Im Beispiel würde man 2.924 finden. Damit ist die Aufgabe gelöst: $\sqrt[3]{25}$ = 2.924

Die oben geschilderten 4 Operationen würde nun Ihr Computer wie folgt lösen:

Schritt 1: Mit ?LOG(25) „sucht" der Computer den Logarithmus für die Zahl 25.
Schritt 2: Das „geteilt durch 3" erledigen Sie mit: ?LOG(25)/3; Sie erhalten 1.Ø7296
Schritt 3: Mit 1.Ø7296 kennt der Computer den Logarithmus des gesuchten Ergebnisses.
Schritt 4: Das Verwandeln des Logarithmus aus Schritt 3 in das gewünschte Ergebnis bewirken Sie mit: ? EXP(1.Ø7296)
Der Computer antwortet Ihnen mit 2.924Ø2, was – wie Sie wissen – richtig ist.
Demnach können Sie die Aufgabe: $\sqrt[3]{25}$ = ? durch den Computer – unter anderem – so lösen:

? EXP(LOG(25)/3)

Ich habe oben „unter anderem" geschrieben, weil ich weiß, daß Sie wohl inzwischen leicht ungehalten sind. Jedenfalls, wenn Sie wissen, daß man $\sqrt[3]{25}$ = ? auch so rechnen kann:

? 25 ↑ (1/3)

Aber es geht hier um LOG... und EXP..., vielleicht haben Sie noch Geduld für den Rest?

Dann beschäftigen Sie sich noch einmal mit Schritt 2!
Obwohl die Aufgabe lautete: „Dritte Wurzel aus 25" haben Sie den Logarithmus von 25 nur durch 3 *geteilt*!

Und darin lag in den rechnerlosen Zeiten der große Vorteil der Logarithmentafel; das Rechnen mit Logarithmen ermöglichte es, jede Rechenart nach den Methoden der nächst niederen Stufe zu lösen.

● Multiplizieren wird zurückgeführt auf Addieren

A * B wird zu ...LOG(A) + LOG(B)...

● Dividieren wird zurückgeführt auf Subtrahieren

A / B wird zu ...LOG(A) − LOG(B)...

Ferner – und das war ganz besonders (rechen-)zeitsparend:

● Potenzieren wird zurückgeführt auf Multiplizieren

A^B wird zu ...LOG(A) * B

● Radizieren wird zurückgeführt auf Dividieren

$\sqrt[B]{A}$ wird zu ...LOG(A) / B

Das folgende kleine Programm führt Ihnen noch einmal die verschiedenen Techniken vor:

```
10 A = 64 : B = 3
20 PRINT A * B ,, EXP(LOG(A) + LOG(B))
30 PRINT A / B ,, EXP(LOG(A) - LOG(B))
40 PRINT A ↑ B ,, EXP(LOG(A) * B)
50 PRINT A ↑ (1/B) ,, EXP(LOG(A) / B)
```

Sie werden erhalten:

192	192
21.3333	21.3333
262144	262144
4	4

Zum Abschluß noch ein wichtiger Tip:

Wenn man Programme für „Geldgeschäfte" (z. B. Zinseszinsrechnung, Berechnung der Annuität usw.) entwirft, dann pflegt man die „Laufzeit" – die Anzahl

der Jahre – mit der Variablen N zu verschlüsseln. Und oft haben Sie dann folgende Nuß zu knacken:

Wie ermitteln Sie N in Beziehungen wie:
$X = A^N$; $\sqrt[N]{X} = A$
Ganz einfach: N = LOG(X) / LOG(A)
Beispiel:

```
10 X = 64 : A = 4
20 PRINT" 64 IST 4 HOCH ";LOG(X) / LOG(A)
```

4.5 Vermischtes

Ein neues Programm!

```
10 A = 3.65 : B = -3.65
20 PRINT"INT(A)=";INT(A),"INT(B)=";INT(B)
30 PRINT"ABS(A)=";ABS(A),"ABS(B)=";ABS(B)
40 PRINT"SGN(A)=";SGN(A),"SGN(B)=";SGN(B)
50 PRINT"FIX(A)=";FIX(A),"FIX(B)=";FIX(B)
```

Der Computer wird Ihnen ausgeben:

INT(A) = 3	INT(B) = −4
ABS(A) = 3.65	ABS(B) = 3.65
SGN(A) = 1	SGN(B) = −1
FIX(A) = 3	FIX(B) = −3

Zu diesen 4 Funktionen liefere ich Ihnen folgende Erklärungen:

Zeile 20:
INT(A) ist die INTeger- oder Ganzzahlfunktion. Diese ist Ihnen aus dem Band „BASIC für Einsteiger" hinreichend bekannt; sie ist hier nur noch einmal zum Vergleich aufgeführt.

Zeile 30:
ABS(A) gibt den ABSolutbetrag des Argumentes, das ist die Zahl *ohne* – positives oder negatives – Vorzeichen.

Zeile 40:
Diese Funktion liefert ein SiGNum – eine Kennzahl, die aussagt, welches Vorzeichen das Argument der Funktion hat.

Ist das Argument *negativ*	ist das SGN	−1
Ist das Argument *positiv*	ist das SGN	1
Ist das Argument *Null*	ist das SGN	0

Zeile 5Ø:
FIX(A) liefert – ähnlich INT... – den Ganzzahl-Anteil des Argumentes. Dabei verzichtet FIX... im Gegensatz zu INT... auf das Abrunden. Anmerkung: FIX(A) ist nicht bei jedem Home-Computer verfügbar!
Schreiben Sie noch ein kleines Übungsprogramm!

```
1Ø A = 7
2Ø PRINT A
3Ø A = A -1.3
```

Sie wissen, daß das eine „Schleife" werden soll? Nun, Sie müßten jetzt Ihrem Computer sagen, wann er mit der Subtraktion aufhören soll. Das ist wegen des „krummen" Ausdrucks gar nicht so einfach!

Ich biete Ihnen für diese – und ähnliche – Fälle folgende Lösung an:

```
4Ø IF SGN(A) = 1 THEN 2Ø
5Ø END
```

Oder:

```
4Ø IF SGN(A) = -1 THEN 6Ø
5Ø GO TO 2Ø
6Ø END
```

Hier lassen Sie also solange rechnen, wie A positiv ist, bzw. befehlen den Abbruch des Spieles, sobald A negativ wird.

4.6 Zusammenfassung

Alle in diesem Abschnitt behandelten Funktionen sind nach dem gleichen Schema aufgebaut:

Funktionsname(A)

Darin ist A das Argument der Funktion, dieses ist stets und ohne Ausnahme in ( ) zu setzen.
Als Argument können Sie verwenden:

- Eine Variable wie in ABS(A)
- Eine Konstante (Zahl) wie in INT(3.4)
- Eine (Standard-)Funktion wie in INT(SQR(A))
- Einen mathematischen Ausdruck wie in FIX(3 * 5/7)

5 Neue Typen von Variablen

Numerische Variable wie A = 1Ø, Z9 = 125 sind Ihnen ebenso geläufig wie Stringvariable A$ = „BASIC" oder Z9$ = „IST SCHOEN". In beiden Fällen ist der Name der Variablen A bzw. Z9.

Das Dollar-Zeichen – $ – ist in BASIC ein „Declaration-Character", ein – frei übersetzt – „Definitionszeichen".

Findet der Computer hinter einem Variablennamen das Zeichen $, dann weiß er, daß er A bzw. Z9 als String zu definieren und im weiteren Verlauf des Programms zu behandeln hat. So weit, so gut.

Es gibt nun Computer, die es erlauben, Variable auch noch anders als oben beschrieben zu definieren. Vorausgesetzt, Ihr Computer verfügt über diese Fähigkeiten, sollten Sie einmal eingeben:

```
5Ø A% = 123.4 : B% = 34.2
6Ø PRINT A%,B%,A% + B%,A% * B%
```

Sie werden dann als Ergebnis erhalten:

```
123   34   157   4182
```

Hier hat also der Computer die Variablen A und B als INT-Variable, als Ganzzahl-Variable behandelt; durch das Definitionszeichen % wurde er darauf aufmerksam gemacht.

Das scheint zunächst wenig Sinn zu machen. Denn Sie hätten ja gleich A = 123, B = 34 sagen können.

Aber haben Sie schon einmal erlebt, daß im Laufe eines Programms mit ganzzahligen Variablen plötzlich noch etwas „hinter dem Komma" erscheint? Ich erinnere mich an einen Fall, bei dem der Computer aus 5↑2 unbedingt 25.ØØØ1 machen wollte. Unglücklicherweise war die Variable in eine IF...THEN...-Entscheidung eingebaut und ich habe lange gesucht, bis ich den vermeintlichen Fehler gefunden hatte.

Durch % können Sie die Bedeutung der Variablen nicht nur am Anfang eines Programms festlegen; Sie können an beliebiger Stelle im Programm neu- oder umdeklarieren.

```
...A = 5
...B = SQR(A)
...C = B * A
```

Und – von hier an:
...D% = C oder auch C% = C
Hier haben Sie also erst in der vierten Programmzeile die Variable C zu einer Ganzzahl-Variablen D% bzw. C% gemacht.

Ihr Computer rechnet bei numerischen Variablen üblicherweise mit 6 Stellen, also

1.23456
123.456
12345.6 usw.

Wenn Ihnen das für bestimmte Rechnungen nicht genau genug ist, können Sie Ihren Computer gewaltig beflügeln. Er wird dann beispielsweise Variable akzeptieren wie 123.4567890123456. In diesem Falle rechnet der Computer mit „doppelter Genauigkeit" (englisch: Double Precision); er liefert Ergebnisse mit 16 Stellen. Zu dieser Leistung beflügeln Sie Ihren Computer mit einem besonderen Definitionszeichen, mit dem Zeichen #

```
10 A# = 123.8 : B# = 34.2
20 PRINT A# * B#
```

Das Ergebnis wird sein: 4233.960198822024

Auch hier können Sie die Deklaration im Laufe eines Programmes wechseln, wie folgendes Beispiel zeigt:

```
10 PI = 3.1416 : R = 50
```

Die Variablen PI und R sind in einfacher Genauigkeit – in „Single Precision" – eingegeben.

```
20 F# = PI * R↑2
30 PRINT F#
```

Sie erhalten: 7854.0048828125

Nehmen Sie an, Sie haben Zeile 30 als Zwischenergebnis ausdrucken lassen und wollen von da an Ihr Programm in der üblichen Art fortsetzen.
Sie können nun neu deklarieren:

```
40 G = F#
50 PRINT G * 2
```

Für die Zeile 50 erhalten Sie 15708; von dort an ist also F wieder „normal".

Es ist empfehlenswert, von der Möglichkeit der doppelten Genauigkeit nur maßvollen Gebrauch zu machen. Weil das Arbeiten mit doppelter Genauigkeit mehr Speicherplatz erfordert und zu höheren Rechenzeiten führt.

6 Sie erweitern das PRINT-Kommando

Für dieses Kommando gibt es – je nach Computertyp bzw. BASIC-„Dialekt" einige Modifikationen; von diesen soll hier die Rede sein.

6.1 PRINT TAB (X) A

Durch Setzen eines Kommas auf einer PRINT-Zeile können Sie bekanntlich den Computer veranlassen, den „Ausdruck" eine Spalte weiter rechts fortzusetzen; bei den meisten Computern sind max. 4 Spalten auf diese Weise möglich.

Mit PRINT TAB (X)... verfügen Sie über die Möglichkeit, den Ausdruck auf jeder der Ø...63 Positionen einer Zeile beginnen zu lassen, eine für das Drucken von Tabellen mit mehr als 4 Spalten sehr nützliche Sache.

„Drucke B auf Position 15" heißt:
PRINT TAB (15) "B"
Beachten Sie, daß hier keinerlei „Satzzeichen" verwendet werden!

Beispiel:

```
5 CLS : A = 1 : B = 2 : C = 3
1Ø PRINT TAB (Ø)"WERT 1";TAB(1Ø)"WERT 2";TAB (2Ø)"WERT 3";TAB(3Ø)
"WERT 4";TAB(4Ø)"WERT 5";TAB(5Ø)"WERT 6";TAB(57)"WERT 7"
2Ø PRINT"=====================================
=====
3Ø PRINT TAB(Ø) A+B;TAB(1Ø)A*B;TAB(2Ø)A*(B+C);TAB(3Ø)A+C;TAB(4Ø)
C-A;TAB(5Ø) C/B;TAB(57)C-B
```

6.2 PRINT @ X,A PRINT AT X,A

Mit dem Kommando PRINT @... kann man *jede Buchstabenposition* des Bildschirmes erreichen. Bei einem Format von 16 Zeilen/64 Zeichen erfolgt dabei die Zählweise wie folgt:

Positionen:

Zeile 1 Ø 63
Zeile 2 64 127
.
.
.
Zeile 15 896 959
Zeile 16 96Ø 1Ø23

„Drucke B auf Position 5ØØ" heißt:

PRINT @ 5ØØ,"B"

Beachten Sie, daß hier für die Positions-Nummer zwar auf die Klammer verzichtet ist; dafür muß nach dem Kommando ein Komma stehen.

Wollen Sie – beispielsweise – Ihren Bildschirm-„Ausdruck" mit dem heutigen Datum schmücken, dann schreiben Sie etwa:

```
5 CLS
1Ø INPUT" DATUM EINGEBEN !";A$
........
1ØØ PRINT @ 47,A$
........
```

Mit PRINT @... können Sie ferner Ihre Programme „benutzerfreundlicher" gestalten; etwa eine Aufforderung zur Dateneingabe deutlich sichtbar in die Mitte des Bildschirmes bringen.

```
5 CLS
1Ø PRINT @ 47Ø, "BITTE NACHEINANDER DIE"
2Ø PRINT @ 53Ø, "VARIABLEN A UND B EINGEBEN"
3Ø INPUT A,B
...........
```

Wenn Sie mit PRINT @... ein paar Übungen machen, dann werden Sie feststellen, daß es ohne ein wenig Zählen gar nicht so einfach ist, die gewünschte Position zu treffen. Nützlich ist dabei entweder ein selbstgefertigtes Raster aus Karopapier, in das die einzelnen Postionen eingezeichnet sind, oder ein „Video Display Worksheet", das man beim Hersteller des Computers kaufen kann.

6.3 PRINT USING...;A

Dieses ist ein sehr leistungsfähiges Kommando, das leider nicht bei jedem Computer „Serie" ist bzw. das nur in manchen „extended BASIC"-Versionen angeboten wird. Grob gesagt, erlaubt dieses Kommando die Gliederung der Ausgabe. Sehen Sie sich's an!

Bei einem Computer stört es gelegentlich, daß seine Ausgabe „linksbündig" erfolgt; nämlich so:

```
123456
123.456
1.23456
12345.6          usw.
```

Ganz besonders dann, wenn es bei dem Ausdruck um's Geld geht, wäre ein Ausdruck in der folgenden Form sicher angenehmer:

```
1ØØ.23
  Ø.5Ø
 47.ØØ           usw.
```

Sie sehen, daß oben:
- alle Kommas untereinanderstehen
- auch vor dem Komma eine Null steht (Ø.5Ø)
- Nach dem Komma ggf. Nullen geschrieben sind (47.ØØ)

Es ist also alles genauso, wie Sie das von einer guten Bankabrechnung gewohnt sind. Und genau dasselbe können Sie mit PRINT USING.. erreichen.

Beim Umgang mit PRINT USING gehen Sie wie folgt vor:

* Sie legen zunächst das gewünschte Ausgabeformat in einem String fest.

Beispiel : A$ ="####.##"
Das bedeutet:
Sie wollen max. vier Stellen vor dem Komma
Sie wollen max. zwei Stellen nach dem Komma
ausdrucken. Ihre größte Zahl liegt also bei 9999.99

Üben Sie einmal!

```
5 CLS
1Ø A$="####.##"
2Ø A = 235.739
3Ø PRINT USING A$;A
```

Der Computer wird ausgeben: 235.74 – womit Sie erkennen, daß PRINT USING.. auch noch automatisch rundet!

Üben Sie weiter!

```
1Ø A$ = "####.##"
2Ø FOR A = Ø TO 2ØØ STEP 2Ø
3Ø PRINT USING A$;A*1.234
4Ø NEXT
```

Sehen Sie, daß die Kommas alle untereinanderstehen?

Und auch die weiteren Eigenschaften von PRINT USING.. können Sie erkennen:

- Gegen seine sonstige Gewohnheit druckt der Computer auch die „führende Null"
- Er füllt die Stellen nach dem Komma mit Null(en) auf, wenn diese Stellen keinen Zahlenwert enthalten
- Was Sie nicht sehen können: hat eine Zahl vor dem Komma weniger Stellen als im String definiert (z. B. 1, 2 oder 3) so füllt der Computer die fehlenden Stellen mit „Blanks" auf.

Aber PRINT USING.. kann noch mehr!

Wählen Sie einmal für das oben stehende Programm folgendes Ausgabeformat:
10 A$ = " **#######.## "
Das Ergebnis ist doch verblüffend!

Der Computer hat den Wert in der ersten Zeile so ausgegeben:
*******0.00 und den in der letzten Zeile so:
*****246.80

Im Klartext: Für jede Stelle links vom Komma, auf der kein Zahlenwert steht, hat er ein * gedruckt. Das ist sicher eine nützliche Schreibe, wenn Sie einmal daran denken, durch Ihren Computer Schecks ausdrucken zu lassen. Erreicht hatten Sie dieses Ausgabeformat durch die beiden ** am Anfang des String A$.

Die Möglichkeiten von PRINT USING.. sind damit noch nicht erschöpft. Schreiben Sie – immer noch für das obige Programm:
10 A$="$$ ###.##" und, nach einem Probelauf:
10 A$="**$ ###.##".
Kommentar ist wohl überflüssig!?

7 Die Behandlung von Strings

In diesem Abschnitt werden wir eine Reihe von Funktionen einüben, mit denen Sie
– mit Strings „Datenverarbeitung" betreiben können und die Ihnen
– Informationen über Strings, ihren Aufbau und ihre Zusammensetzung, liefern.
Grundlage dafür ist der sogenannte ASCII-Code, nach dem alle Ihnen über die Tastatur zugänglichen Ziffern, Buchstaben und Zeichen für den Computer verschlüsselt werden.

„ASCII" steht dabei für „American Standard Code for Information Interchange", was, frei übersetzt, heißt: „Amerikanischer Standardcode für Informationsaustausch".

In diesem Code werden für Mikrocomputer alle Zeichen auf der Tastatur durch den sogen. „Character Generator" in acht Bit breite Datenworte umgewandelt, die im Bereich ØØØØØØØØ...11111111 liegen. Ins Dezimale übersetzt liegt also der Code im Bereich Ø...255; somit können 256 Zeichen verschlüsselt werden.

Tabelle 4: Übersicht über den ASCII-Code

Code	Zeichen/Funktion
0–7	keine
8	ein Zeichen rückwärts und Löschen des Zeichens
9	keine
10–13	„Wagenrücklauf"; "ENTER"
14	Einschalten Cursor
15	Ausschalten Cursor
16–22	keine
23	Übergang auf 32 Zeichen/Zeile
24	Cursor eine Position zurück
25	Cursor eine Position vorwärts
26	Cursor nach unten; neue Zeile
27	Cursor nach oben
28	Cursor an Anfang 1. Zeile, Position Ø,Ø
29	bewegt Cursor an Anfang der Zeile
30	löscht bis Ende der Zeile
31	Rücksetzen des Speicherinhaltes auf Ø (Clear)

Code	Zeichen/Funktion	Code	Zeichen/Funktion
32	Zwischenraum, „Blank“	69	E
33	!	70	F
34	”	71	G
35	#	72	H
36	$	73	I
37	%	74	J
38	&	75	K
39	’	76	L
40	(	77	M
41	)	78	N
42	*	79	O
43	+	80	P
44	’	81	Q
45	–	82	R
46	.	83	S
47	/	84	T
48	0	85	U
49	1	86	V
50	2	87	W
51	3	88	X
52	4	89	Y
53	5	90	Z
54	6	91	↑oder [
55	7	92	↓
56	8	93	←
57	9	94	→
58	:	95	–
59	;	96–127	Kleinbuchstaben der Zeichen 64–95
60	<		
61	=	128	Zwischenraum, „Blank“
62	>	129–191	Graphische Zeichen
63	?	191–255	TAB-Funktionen
64	@		
65	A		
66	B		
67	C		
68	D		

Tabelle 4 zeigt Ihnen eine Übersicht über den ASCII-Code, zu dem es einige Bemerkungen zu machen gibt:

- Dargestellt ist in der Tabelle der gesamte Code mit 256 Zeichen. Nicht jeder Mikrocomputer verfügt über den vollen Zeichensatz; manche begnügen sich etwa mit der Hälfte (127 Zeichen). Sie werden gleich ein Programm kennenlernen, mit dem Sie herausfinden können, was Ihre Maschine zu bieten hat.
- Sie sehen aus der Tabelle, daß z. B. den Code-Nummern 48...57 die Ziffern Ø...9, den Code-Nummern 65...9Ø die Großbuchstaben A...Z entsprechen.
- Sie sehen ferner, das die „größere" Ziffer die höhere Code-Nummer und der „größere" Buchstabe ebenfalls die höhere Code-Nummer hat.

  5 hat die Code-Nummer 53
  6 hat die Code-Nummer 54
  X hat die Code-Nummer 88
  Y hat die Code-Nummer 89 usw.

Dieser Zusammenhang zwischen Zeichen und ASCII-Code-Nummer ist besonders wichtig; er ist die Grundlage für die Fähigkeit des Computers, „Datenverarbeitung" wie z. B. das Sortieren von Ziffern und Zeichen zu betreiben.

7.1 ASC(...) ; CHR$(...)

Diese beiden Funktionen erlauben Ihnen, mit ASCII-Code-Nummern auf der einen und den zugehörigen Zeichen auf der anderen Seite zu „spielen". Geben Sie einmal ein:

Ihre Eingabe...	*...die Computerausgabe*
PRINT ASC("B")	66
PRINT ASC(B)	? TM ERROR
PRINT ASC("?")	63

Wenn Sie jetzt Tabelle 4 zu Rate ziehen, werden Sie die Computerausgabe leicht deuten können. In Ihrer ersten Eingabe hatten Sie für das Argument der ASC-Funktion korrekt "B" eingegeben. Der Computer hat mit dem ASCII-Code für B, mit 66, geantwortet.

Demnach liefert diese Funktion den ASCII-Code für einen eingegebenen String.

- Das Argument muß korrekt in " " stehen! Das hatten Sie bei der zweiten Eingabe übersehen!

– Das Argument der Funktion darf nur aus einem Zeichen bestehen! Bei PRINT ASC("BASIC") erfolgt je nach System entweder eine Fehlermeldung oder als Antwort schlicht: 66; und das ist die Code-Nummer für B, für den ersten Buchstaben des von Ihnen eingegebenen Argumentes.

Versuchen Sie eine neue Eingabe:

Ihre Eingabe...	*...die Computerausgabe*
PRINT CHR$(66)	B
PRINT CHR$(51)	3
PRINT CHR$(63)	?

Sie erkennen, daß es sich bei CHR$.. um die Umkehrung der ASC-Funktion handelt:

CHR$(...) liefert den String für die in ( ) eingegebene ASCII-Code-Nummer.

Als Codenummer darf nur eine Zahl zwischen Ø und 255 eingegeben werden.

Und hier das angekündigte Programm zur Durchmusterung Ihres „Character Generators":

```
1Ø CLS
2Ø INPUT"AB WELCHER CODE-NUMMER ?";A
3Ø INPUT"BIS ZU WELCHER CODE-NUMMER ?";B
4Ø PRINT"CODE-NUMMER","ASCII-ZEICHEN"
5Ø PRINT"==========================================
=====
6Ø FOR I = A TO B
7Ø PRINT I,CHR$(I)
8Ø NEXT
```

Beginnen Sie mit A = 33 und wählen Sie B = A + 1Ø !! Wenn Ihr Computer für Grafik-Betrieb eingerichtet ist, werden ab A = 127 die verschiedenen grafischen Zeichen ausgegeben. Viel Spaß bei der Musterung!

7.2 LEN (...)

Mit dieser Funktion erfahren Sie die „Länge" (englisch: *LEN*GTH für Länge) des in ( ) stehenden String; der Computer verrät Ihnen, aus *wievielen Zeichen* der String besteht.

Zur Übung:

Ihre Eingabe...	*...die Computerausgabe*
PRINT LEN("BASIC")	5
1Ø A$ = "BASIC"	
2Ø PRINT LEN(A$)	5
PRINT LEN("BASIC COMPUTER")	14

Bei Ihrer ersten Eingabe war die Antwort 5, weil der String BASIC aus 5 Zeichen besteht.

Bei Ihrer zweiten Eingabe wurde die Form des Argumentes verändert; Sie können also entweder einen String im „Klartext" – wie „BASIC" – eingeben oder einen bereits mit A$ verschlüsselten String.

Und bei dem dritten Beispiel bitte ich Sie, einmal genau zu zählen! Die Ausgabe lautete: 14!! Auch der Zwischenraum (= ein „Blank") zwischen BASIC und COMPUTER wird als Zeichen gezählt!

7.3 Stringausgaben

● **LEFT$(STRING,N)**

In dieser Funktion bedeutet LEFT soviel wie „Links"; ausgegeben werden N Zeichen, beginnend am *linken Rand* des als Argument der Funktion eingesetzten String. Das wollen wir wieder probieren!

Ihre Eingabe...	*...die Computerausgabe*
PRINT LEFT$("BASIC",1)	B
1Ø A$="BASIC"	
2Ø PRINT LEFT$(A$,1)	B
3Ø PRINT LEFT$("COMPUTER",3)	COM

Das erklärt sich fast von selbst; Sie sollten anhand des folgenden Programms einmal eine Nutzanwendung dieser Funktion untersuchen!

```
5 CLS
1Ø READ A$
2Ø IF LEFT$(A$,2) = "BA" THEN 5Ø
3Ø GO TO 1Ø
5Ø PRINT A$
6Ø GO TO 1Ø
7Ø DATA "BACH", "BAMMEL", "BUSCH", "BAUMANN", "BIRNE", "BALKEN"
```

Der Computer wird Ihnen ausgeben:

BACH
BAMMEL
BAUMANN
BALKEN

Hier wurde also mit Hilfe der LEFT$-Funktion aus einer Liste von Strings nacheinander jeder String herausgesucht – und aufgeschrieben – dessen *erste beiden Zeichen = LEFT*$(STRING,N) BA sind.

Bewerkstelligt haben Sie das mit der Zeile 2Ø; Sie lernen daraus, daß man mit den hier behandelten Funktionen auch „arbeiten" kann!

● **RIGHT$(STRING,N)**

In dieser Funktion bedeutet RIGHT soviel wie „Rechts"; ausgegeben werden N Zeichen, beginnend am *rechten Rand* des als Argument der Funktion eingesetzten STRING. Unsere Probe:

Ihre Eingabe...	*...die Computerausgabe*
PRINT RIGHT$("BASIC",1)	C
1Ø A$="BASIC"	
2Ø PRINT RIGHT$(A$,1)	C
PRINT RIGHT$("BASIC",3)	SIC

Ein kleines Übungsprogramm:

```
5 CLS : A$="BASIC"
1Ø FOR N = 1 TO 5
2Ø PRINT RIGHT$(A$,N)
3Ø NEXT N
```

Der Computer gibt aus:

```
C
IC
SIC
ASIC
BASIC
```

● **MID$(STRING,N,M)**

In dieser Funktion bedeutet MID soviel wie „Mitte" (englisch: *MID*DLE); ausgegeben wird ein *TEIL*-STRING; und zwar:

* beginnend beim *N*-ten Zeichen
* mit einer Länge von *M* Zeichen.

Ihre Eingabe...	*...die Computerausgabe*
PRINT MID$("BASIC",3,2)	SI
PRINT MID$("BASIC",3,1)	S
1Ø A$="BASIC"	
2Ø PRINT MID$(A$,2,3)	ASI

Lassen Sie sich einmal von Ihrem Computer folgendes auf den Schirm bringen; Sie werden dann die Funktion ganz verstehen!

```
5 CLS : A$="BASIC"
1Ø FOR M = 1 TO 5
2Ø FOR N = 1 TO 5
3Ø PRINT MID$(A$,N,M)
4Ø NEXT N
5Ø NEXT M
```

7.4 Die Codier- und Decodiermaschine

Falls Sie einmal Geheimagent werden wollen, dann müssen Sie Botschaften in verschlüsselter Form senden und verschlüsselt empfangene wieder entschlüsseln, damit Sie diese verstehen können. Die bis jetzt behandelten String-Funktionen versetzen Sie in die Lage, ein Programm zu schreiben, das Ihren Computer in so ein Geheimdienstwerkzeug umfunktioniert. Beginnen wir mit dem Programm, das Ihren „Klartext" verschlüsselt!

```
  5 CLS
 10 CLEAR 1000
 20 PRINT "GEBEN SIE DIE ZU CODIERENDE NACHRICHT EIN!"
 25 INPUT N$
 30 FOR I = 1 TO LEN (N$)
 40 X$ = MID $ (N$, I, 1)
 50 C = ASC (X$) + 5
 60 IF C > 255 THEN C = C - 255
 70 C$ = C$ + CHR $ (C)
 80 NEXT I
 90 PRINT "DIE CODIERTE NACHRICHT LAUTET:"
100 PRINT C$
```

Geben Sie in dieses Programm BASIC ein, dann wird der Computer das zu GFXNH „verschlüsseln". Dies geschieht offensichtlich in den Programmzeilen 30...80. Sie kommen hinter die Raffinessen des Programms, wenn Sie einmal folgende „Hilfszeilen" eingeben:

```
45 PRINT "X$ ="; X$,
55 PRINT "C ="; C,
75 PRINT "C$ ="; C$,
76 PRINT "C$ + CHR $ (C) ="; C$ + CHR $ (C)
```

Ergebnis:

X$ = B	C = 71	C$ = G	C$ + CHR $ (C) = GG
X$ = A	C = 70	C$ = GF	C$ + CHR $ (C) = GFF
X$ = S	C = 88	C$ = GFX	C$ + CHR $ (C) = GFXX
X$ = I	C = 78	C$ = GFXNF	C$ + CHR $ (C) = GFXNN
X$ = C	C = 72	C$ = GFXNF	C$ + CHR $ (C) = GFXNHH

DIE CODIERTE NACHRICHT LAUTET:
GFXNH

Der „Trick" liegt offenbar in der Zeile 50, in der Sie zu der ASC-Nummer einfach 5 addieren! Daß dem so ist, erfahren Sie wie folgt:

Ändern Sie für den zweiten Versuch zusätzlich:
50 C = ASC (X$)
EINGABE in 20 wieder BASIC

Ergebnis:

X$ = B	C = 66	C$ = B	C$ + CHR $ (C) = BB
X$ = A	C = 65	C$ = BA	C$ + CHR $ (C) = BAA
X$ = S	C = 83	C$ = BAS	C$ + CHR $ (C) = BASS
X$ = I	C = 73	C$ = BASI	C$ + CHR $ (C) = BASII
X$ = C	C = 67	C$ = BASIC	C$ + CHR $ (C) = BASIC

DIE CODIERTE NACHRICHT LAUTET:
BASIC

Hier erfolgt also keine Verschlüsselung Ihrer Eingabe!
Das Programm für die Entschlüsselung:

```
  5 CLS
 10 CLEAR 1000
 20 PRINT "GEBEN SIE DIE CODIERTE NACHRICHT EIN!"
 30 INPUT N $
 40 FOR I = 1 TO LEN (N$)
 50 X$ = MID $ (N$, I, 1)
 60 C = ASC (X$) - 5
 70 IF C < 0 THEN C = C + 255
 80 D$ = D$ + CHR $ (C)
 90 NEXT
100 PRINT "DIE DECODIERTE NACHRICHT LAUTET:"
110 PRINT D$
```

Eingabe: GFXNH
Ausgabe: BASIC
Dieses Programm entschlüsselt die codierte Nachricht also durch die *Subtraktion* von 5! (Zeile 60!)

7.5 Welcher String steckt im String?

Mit dem folgenden Übungsprogramm können Sie untersuchen, ob in einem bekannten String ein bekannter *Teil*-String enthalten ist. Sie könnten fragen: Kommt in dem String „Waldmeister" der Teilstring „Meister" vor?

```
  5 CLS : CLEAR 1ØØØ
 2Ø INPUT"GEBEN SIE DEN STRING EIN !";A$
 3Ø INPUT"GEBEN SIE DEN TEILSTRING EIN !";B$
 4Ø GOSUB 3ØØ
 5Ø IF I = Ø THEN 1ØØ
 6Ø PRINT B$;" IST TEIL DES STRING ";A$
 7Ø PRINT"BEGINN BEIM ";I;" TEN BUCHSTABEN"
 8Ø PRINT"ENDE BEIM ";I+LEN(B$)-1;" TEN BUCHSTABEN"
 9Ø GOTO 2Ø
1ØØ PRINT B$;" IST NICHT IN ";A$;" ENTHALTEN"
11Ø GOTO 2Ø
3ØØ FOR I = 1 TO LEN(A$)-LEN(B$)+1
31Ø IF B$=MID$(A$,I,LEN(B$)) THEN RETURN
32Ø NEXT I
33Ø I=Ø
34Ø RETURN
```

Bei diesem Übungsprogramm kommt es nicht auf das Ergebnis an; ich habe es hingeschrieben, um Ihnen zu zeigen, was Sie mit diesen Funktionen anstellen können und wie man sie korrekt hinschreibt; erinnern Sie sich daran, was Sie vor wenigen Seiten bei der Behandlung mathematischer Funktionen gelernt haben! Betrachten Sie einmal Zeile 3ØØ: Hier haben Sie eine FOR...NEXT...Schleife eingesetzt, deren „Länge" von 1 reicht bis zu: Länge des String A$ MINUS Länge des String B$+1!
Sie können also dann mit den String-Funktionen „rechnen", wenn die Funktion eine Zahl ausgibt (das tut, wie Sie wissen, LEN( )!!).
Oder übersetzen Sie einmal die Zeile 31Ø „ins Deutsche"! Die Grundfunktion lautet: MID$(STRING,N,M)

- STRING ist A$; das ist der (lange) String, dessen Eingabe das Programm von Ihnen in Zeile 2Ø erwartet
- N ist in Zeile 31Ø gleich I; kann also entsprechend der in Zeile 3ØØ begonnenen Schleife nacheinander die Werte 1,2... annehmen, solange, bis M erreicht ist
- M ist in Zeile 31Ø gleich LEN(B$)+1; B$ ist der (Teil-)String, den Sie in Zeile 3Ø eingegeben haben.

Zeile 31Ø durchsucht also den String A$ Zeichen für Zeichen des *String B$*. Zur weiteren Übung schlage ich Ihnen vor, das letzte Programm noch um folgende Zeilen zu erweitern:

```
82 PRINT"DER STRING ";A$;" HAT";LEN(A$);" ZEICHEN"
84 PRINT"DER STRING ";B$;" HAT";LEN(B$);" ZEICHEN"
86 PRINT"DER STRING ";A$;" IST";LEN(A$)-LEN(B$);" ZEICHEN LAENGER
   ALS ";B$
```

7.6 STRING$(N,...)

Diese nicht bei jedem Personalcomputer verfügbare Funktion
– gibt einen String aus, der
– aus N Zeichen
– eines „String" oder einer Code-Nummer
besteht.

Auf PRINT STRING$(5,"A") erhalten Sie also: AAAAA

Auf PRINT STRING$(5,65) erhalten Sie ebenfalls AAAAA, weil 65 die Code-Nummer für „A" ist.

Aus der Tabelle 4 können Sie entnehmen: Die Code-Nummer für * ist 42. Die Code-Nummer für = ist 61 usw.

Wollen Sie also etwa einen Tabellen-Kopf durch eine ganze Zeile mit Gleichheitszeichen „unterstreichen", dann brauchen Sie mit Hilfe der neuen Funktion nur eingeben:

...PRINT STRING$(64,61)

64 deshalb, weil eine Zeile 64 Zeichen lang ist und 61 deshalb, weil 61 die Code-Nummer für = ist.

Wohl dem, der's hat!

7.7 VAL(STRING)

Diese Funktion liefert den Zahlenwert des Argumentes; VAL steht dabei für das englische Wort für *VAL*ue = „Wert".

Das werden Sie nicht auf Anhieb verstehen, deshalb wollen wir das an einem kleinen Programm einüben!

Ihre Eingabe...	*...die Computerausgabe/Erklärung*
? VAL(3Ø)	? TM ERROR Das Argument muß ein *String* sein!
? VAL("3Ø")	3Ø So ist's recht! "3Ø" ist ein String!
? VAL("BASIC")	Ø "BASIC" ist ohne Zweifel ein korrekt eingegebener String. "BASIC" hat aber *keinen Zahlenwert*; der Computer gibt dafür Ø aus!
? VAL("3Ø DM")	3Ø Der Computer gibt den Zahlenwert (oder „Anteil") des String "3Ø DM" aus!
? VAL("DM 3Ø")	Ø (Erklärung siehe unten!)
? VAL("31.12.83")	31.12

An den beiden letzten Beispielen erkennen Sie, daß VAL den String Zeichen für Zeichen daraufhin untersucht, ob er – der String – als Zahl zu interpretieren ist. Bei "DM 3Ø" hat er bereits bei "D... erkannt, daß D keinen Zahlenwert hat, und das Handtuch geworfen. Das heißt: Er hat die weitere Suche abgebrochen und Ø ausgegeben.

Bei VAL("31.12.83") hatten Sie als String ein Datum eingegeben. VAL... interpretiert das bis 31.12 als Zahl. Schließlich kann ja auch "." *Teil* einer Zahl sein wie eben in 31.12...

Aber da es keine Zahl mit „zwei Kommas" gibt, hat der Computer bei Erkennen des zweiten Kommas die Suche abgebrochen und nur das bis dahin Erkannte ausgegeben (31.12).

Ähnlich hat er es übrigens oben mit 3Ø DM gemacht!

Vertiefen Sie Ihr Wissen um VAL... mit einem kleinen Programm!

```
1Ø A$="1ØØ STUECK"
2Ø B$=" 1Ø DM/STUECK"
3Ø PRINT"GESAMTWERT= ";VAL(A$)*VAL(B$);" DM"
```

Das Ergebnis dieser Operation ist: GESAMTWERT = 1ØØØ DM

7.8 STR$(...)

Diese Funktion verwandelt die in ( ) stehende Zahl oder numerische Variable in einen String um; nach der Umwandlung können Sie also etwa die Zahl 135 nach den bisher bekannten Regeln als String weiterverarbeiten.

```
1Ø A = 1ØØ
2Ø B = 1Ø
3Ø A$ = "STUECK"
4Ø B$ = "DM/STUECK"
```

Wandeln Sie nun A und B in einen String um! Das geht so:

```
5Ø C$ = STR$(A)
6Ø D$ = STR$(B)
```

Und nun werden die Strings „verarbeitet", miteinander verknüpft:

```
7Ø PRINT"LAGERBESTAND =";C$+A$
8Ø PRINT"EINZELPREIS =";D$+B$
9Ø PRINT"GESAMT-LAGERWERT =";VAL(C$)*VAL(D$);" DM"
```

Sie erhalten:

Zeile 70: LAGERBESTAND = 100 STUECK

Zeile 80: EINZELPREIS = 10 DM/STUECK

In den Zeilen 70 und 80 wurden durch das +–Zeichen die Strings jeweils nur aneinandergehängt; nacheinander hingeschrieben.
Man nannt das „Concatenation"

Zeile 90: GESAMT-LAGERWERT = 1000 DM

Hier wurden die *Zahlenwerte* (VAL!) der Strings C$ und D$ miteinander *multipliziert.*

7.9 INKEY$, GET

Diese beiden Funktionen erlauben die Eingabe eines nur aus einem Zeichen bestehenden Strings während der fortlaufenden Abtastung der Tastatur durch den Computer. INKEY.. und GET... erfüllen die gleichen Aufgaben. Ihr Computer hat aber entweder die eine oder die andere Funktion, so daß wir diese nacheinander behandeln wollen.

Beginnen wir mit INKEY...

Durch den Computer wird Ihre Eingabetastatur fortlaufend abgetastet. Sobald eine Taste gedrückt ist, erkennt das Ihr Apparat, verschlüsselt die gedrückte Taste und gibt das entsprechende Zeichen etwa auf dem Bildschirm aus. So können Sie Zeichen für Zeichen etwa eine Befehlszeile aufbauen. Erst wenn Sie die Zeile durch Druck auf die Taste ENTER bzw. RETURN abschließen, übernimmt der Computer die Zeile zur weiteren Verarbeitung. INKEY setzt diese Ihnen geläufige Arbeitsweise außer Kraft!

Geben Sie ein:

```
...
100 PRINT"WENN SIE WEITERMACHEN WOLLEN, DRUCKEN SIE DIE TASTE
'J' "
110 INPUT A$
120 IF A$ ="J" THEN 150
130 PRINT"WARUM DENN NICHT ?":END
150 PRINT"DANN FAHREN WIR IM PROGRAMM FORT !"
160.....
.....
```

Wenn Sie dieses Programm starten – und weitermachen wollen – dann müssen Sie eingeben:

J
danach ENTER bzw. RETURN!
Nun: Durch INKEY können Sie auf ENTER bzw. RETURN verzichten, wenn Sie wie folgt vorgehen:

```
100 bleibt wie oben!
110 A$ = INKEY$
120 IF A$=""THEN 110
130 IF A$ = "J" THEN 160
140 PRINT"WARUM DENN NICHT ?"
150 END
160 PRINT"DANN FAHREN WIR IM PROGRAMM FORT !"
```

So; und nun brauchen Sie nur auf die Taste J zu drücken, und augenblicklich erscheint Zeile 160 auf dem Schirm.

Hier bedarf wohl die Zeile 120 noch einer Erklärung. Sie müssen wissen, daß die Tastaturabfrage durch den Computer mit „affenartiger" Geschwindigkeit erfolgt; ehe Sie sich versehen, hat der Computer entdeckt, daß Sie „Nichts" gedrückt haben und interpretiert ihre lange Leitung als bewußte Eingabe von „Nichts"... und das ist eben "". Zeile 120 weist den Computer also an, solange im Kreis zwischen 110 und 120 herumzusuchen, bis Sie (irgendeine) Taste gedrückt haben. Ist das die Taste J, dann geht's weiter im Programm. Ist das eine andere Taste, fragt der Computer WARUM DENN NICHT?

Nun zu GET...

Verfügt Ihr Computer über diese Funktion, dann müßte die Zeile 110 lauten:

```
110 GET A$
```

Haben Sie etwa eine APPLE-Maschine, dann können Sie die Zeile 120 noch einsparen. Weil dieser Computer bei GET automatisch „Warteschleife fliegt", bis eine Taste gedrückt wird.

Damit hätten wir das Kapitel „String-Funktionen" abgeschlossen. Sie sollten nun daran gehen, Ihr zunächst noch reichlich theoretisches Wissen anhand der folgenden Beispiele einzuüben, damit Sie die neuen Funktionen auch anwenden können. Ich habe mich bemüht, Beispiele zu wählen, die Ihnen nicht nur den Einsatz der String-Funktionen erläutern, sondern Ihnen auch nützliche (Unter-) Programme für andere Aufgaben vorführen.

8 Beispiele für String-Manipulationen

8.1 Ordnen von Zahlenkolonnen

Erinnern Sie sich noch an die Beschreibung der PRINT USING-Kommandos? Wenn Sie dort der Neid gepackt hat – weil Ihre Maschine diesen Komfort nicht bietet – dann lernen Sie jetzt ein Programm mit dem Sie – wenn auch mit mehr Programmieraufwand – den gleichen Effekt erreichen können:

```
  5 CLS
  7 D$ = "DM"
 1Ø FOR A = 95.7 TO 105.345 STEP 1.045
 2Ø A = INT (A * 1ØØ + .5) / 1ØØ

 3Ø A$ = STR $ (A)
 4Ø FOR I = 1 TO LEN (A$)
 5Ø IF MID $ (A$, I, 1) = " . " THEN 9Ø
 6Ø NEXT I
 7Ø A$ = A$ + " .ØØ "
 8Ø GO TO 1ØØ
 9Ø IF I = LEN (A$) – 1 THEN A$ = A$ + " Ø "

1ØØ FOR X = 1 TO 2Ø – I
11Ø A$ = "  " + A$
12Ø NEXT X
13Ø PRINT A$ ; D$

14Ø NEXT A
```

Variante: Einfügen: 35 GOTO 13Ø

Das bewirkt, daß A$, D$ *linksbündig* ausgegeben werden. Man erkennt deutlich den Unterschied zu den in 3Ø...12Ø geordneten Zahlen.

Wenn Sie als weitere Variante eingeben:

```
15 PRINT A;D$,,
```

erreichen Sie, daß ungeordnete und geordnete Kolonne zum Vergleich nebeneinander stehen.

Mit dem folgenden Programm soll das Ordnen noch vertieft werden. Es ist für den PC 100 geschrieben und berücksichtigt dessen Drucker mit nur 20 Zeichen/ Zeile; Sie können es aber leicht auf andere Computer abwandeln:

1Ø INPUT A	A ist die Zahl, die geordnet werden soll.
2Ø A$=STR$(INT(A*1ØØ+.5))	A wird in den String A$ umgewandelt und gleichzeitig gerundet.
3Ø PRINT TAB(19 − LEN(A$));	TAB 19 bewirkt rechtsbündigen Ausdruck
4ØPRINT LEFT$(A$,LEN(A$)−2);	Ist A$ z. B. 123.45, dann bewirkt Zeile 4Ø den Ausdruck von 123
5Ø PRINT".";RIGHT$(A$,2)	Der erste Teil von Zeile 5Ø bewirkt Ausdruck des „Kommas"; der zweite Teil bewirkt Ausdruck von 45

Da die Befehle 3Ø, 4Ø, 5Ø durch den Abschluß der Zeile mit ; auf *einer Zeile* gedruckt werden, ist das Ergebnis so, wie Sie es wünschen.

Bleiben wir noch bei diesem Programm! Schließlich wollen Sie ja nicht nur Kolonnen mit Zahlen, die zwei Stellen hinter dem Komma haben, „kommazentriert" schreiben, sondern Zahlen mit beliebig vielen Stellen nach dem Komma! Also:

```
 5 INPUT"WIEVIELE STELLEN NACH DEM KOMMA SOLL DIE AUSGABE
   HABEN ?";N
1Ø INPUT"GEBEN SIE DIE ZU ORDNENDE ZAHL EIN ";A
2Ø A$=STR$(INT(A*1Ø↑N+.5))
3Ø PRINT TAB(19−LEN(A$))
4Ø PRINT LEFT$(A$,LEN(A$)−N);
5Ø PRINT".";RIGHT$(A$,N)
6Ø GO TO 1Ø
```

Wenn Sie das einmal in einem „richtigen" Programm benötigen, dann machen Sie aus den Zeilen 2Ø, 4Ø, 5Ø einfach ein Unterprogramm.

8.2 Sie fügen das Datum in Ihr Programm ein

Immer, wenn ich von einem Programm aufgefordert werde, das Datum des Tages einzugeben, beginne ich mich zu ärgern. Weil der Computer mich zwingt, das Datum in einer Form einzugeben, die ich nicht gewohnt bin. Den 1.3.83 will er serviert haben als 3.1.83 oder als Ø1.Ø3.83 oder er fragt blöd daher: MMTTJJ? Und

Beispiel	String-Positionen								String-Länge
	1	2	3	4	5	6	7	8	
1	1	.	1	.	8	3			6
2	1	0	.	1	.	8	3		7 ⓑ
3	1	.	1	0	.	8	3		7 ⓐ
4	3	1	.	1	2	.	8	3	8

Abb. 8 Mögliche Länge der Strings im Datum-Programm

das alles, weil die Programme meist aus den USA kommen und dort die Leute den Monat vor dem Tag schreiben!

Wir wollen uns folgende Aufgabe stellen:
- Das Datum wird in der hierzulande üblichen Form eingegeben:
  12.12.83
- Der Computer soll es umwandeln und weiterverarbeiten als:
  12. Dezember 1983

Weil das Datum eine Zahl „mit zwei Kommas" darstellt und es so etwas nicht gibt, müssen wir die Umwandlung als String-Manipulation betreiben. Die Länge, die unser String annehmen kann, ist von dem jeweiligen Datum abhängig; der 1.1.83 ergibt einen kürzeren String als der 31.2.85 usw. Wieviele Möglichkeiten es gibt, macht *Abb. 8* deutlich.

Nun lösen wir unsere Aufgabe schrittweise; wir gehen zunächst davon aus, daß das Datum etwa 12.12.81 lautet; der String somit acht Zeichen lang ist. Wir verwenden ferner den Variablen-Namen T für Tag; M für Monat und J für Jahr. Das umzuwandelnde Datum geben wir unter D$ ein.

```
100 INPUT D$
105 IF LEN(D$)=8 THEN 140
110 GOTO 100
140 T= VAL(LEFT$(D$,2))
150 M=VAL(MID$(D$,4,2))
160 J=VAL(RIGHT$(D$,2)) : J=J+1900
170 PRINT T;".";M;".";J
```

Damit führt die Eingabe für:

D$=17.10.82

zu der Ausgabe:

17.10.1982

Wir haben unser Ziel knapp verfehlt! Immerhin haben Sie erkannt, wie das Umwandlungsprinzip funktioniert. Setzen Sie das Programm fort für Strings D$ mit sechs bzw. sieben Zeichen Länge!

```
100 PRINT"DATUM EINGEBEN !"
105 INPUT D$
120 IF LEN(D$) = 8 THEN 260
130 IF LEN (D$) = 7 AND MID$(D$,2,1) = "." THEN 310
140 IF LEN (D$) = 6 THEN 360
150 T = VAL(LEFT$(D$,2))
160 M= VAL(MID$(D$,4,1))
170 GO TO 500
260 T=VAL(LEFT$(D$,2))
270 M=VAL(MID$(D$,4,2))
280 GO TO 500
310 T=VAL(LEFT$(D$,1))
320 M=VAL(MID$(D$,3,2))
330 GO TO 500
360 T=VAL(LEFT$(D$,1))
370 M=VAL(MID$(D$,3,1))

500 J=VAL(RIGHT$(D$,2))
510 J=J+1900
520 PRINT T;".";M;".";J
```

Wenn Sie von der Tatsache absehen, daß die Monate immer noch nicht im gewünschten Klartext ausgegeben werden, so ist dieses Programm doch in der Lage, alle Strings der möglichen Länge korrekt umzuwandeln, was Sie ruhig an ein paar Beispielen überprüfen sollten.

Für den Rest des Problems erinnern wir uns der Tatsache, daß ja bereits in unserer Umgangssprache gewissermaßen die „1" ein „Codewort" für Januar, die „5" eines für Mai ist usw. Wir veranlassen unser Programm, immer dann „Januar" bzw. „Oktober" hinzuschreiben, wenn für M die „Codenummer" 1 bzw. 10 auftritt. Dazu legen wir die Monatsnamen in einer Datentabelle am Ende des Programms ab und ergänzen das vorige Programm noch um einen „Leseteil":

```
 4 CLS : DIM MO$(12)
 5 FOR M = 1 TO 12
10 READ MO$(M)
30 NEXT M
```

Die Zeile 52Ø wird abgeändert zu:

```
52Ø PRINT T;".";MO$(M);J
```

Und die Monatsnamen fügen Sie wie folgt ins Programm ein:

```
6ØØ DATA "JANUAR","FEBRUAR","MAERZ","APRIL","MAI","JUNI","JULI"
61Ø DATA "AUGUST","SEPTEMBER","OKTOBER","NOVEMBER",
    "DEZEMBER"
```

Und nun ist wohl der Job zufriedenstellend erledigt?

8.3 Sie verwandeln eine Dezimalzahl in eine Binärzahl

Die Dezimalzahl 16 schreibt man in binärer Darstellung als 1ØØØØ. Oft finden Sie statt 1ØØØØ die Darstellungsart HLLLL. Darin steht an Stelle der 1 der Buchstabe H; das bedeutet *H*IGH für „hoher Pegel" und an Stelle der Ø der Buchstabe L; das bedeutet *L*OW für „niedriger Pegel".

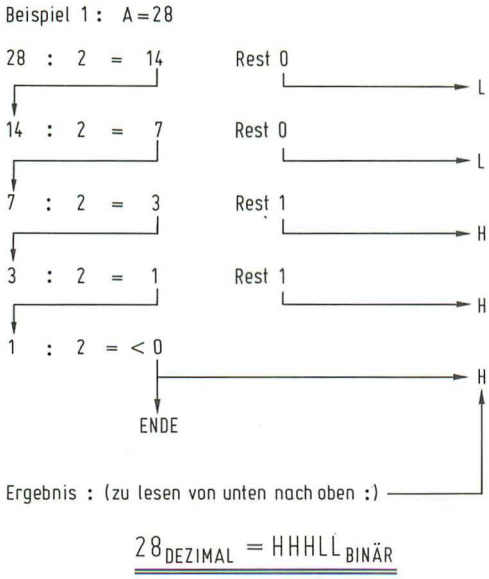


Abb. 9 Umwandlung einer Dezimalzahl in eine Binärzahl (Beispiel 1)

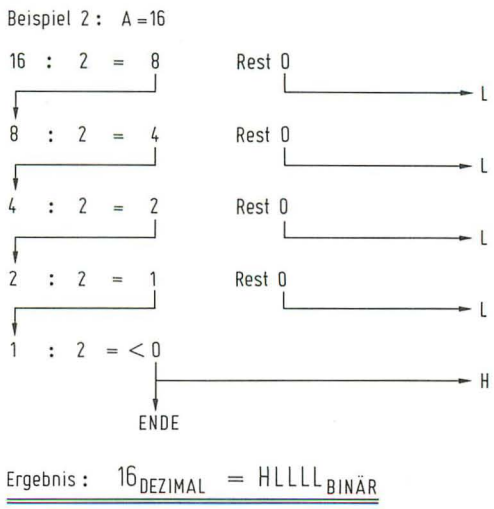


Abb. 10 Umwandlung einer Dezimalzahl in eine Binärzahl (Beispiel 2)

Es kommt nun gelegentlich vor, daß man die eine Darstellungsart in die andere übersetzen will, also eine Dezimalzahl in binärer Form hinschreiben möchte. Das kann man ziemlich mühsam nach alten Regeln – die Sie z. B. in Büchern über Digitaltechnik finden – „zu Fuß" machen; in den *Abb. 9* und *10* habe ich Ihnen das Prinzip aufgemalt. Danach wird die Dezimalzahl fortlaufend durch 2 geteilt. Je nachdem, ob dabei ein „Rest" auftritt oder nicht, ergibt sich das Ergebnis zu L oder H.

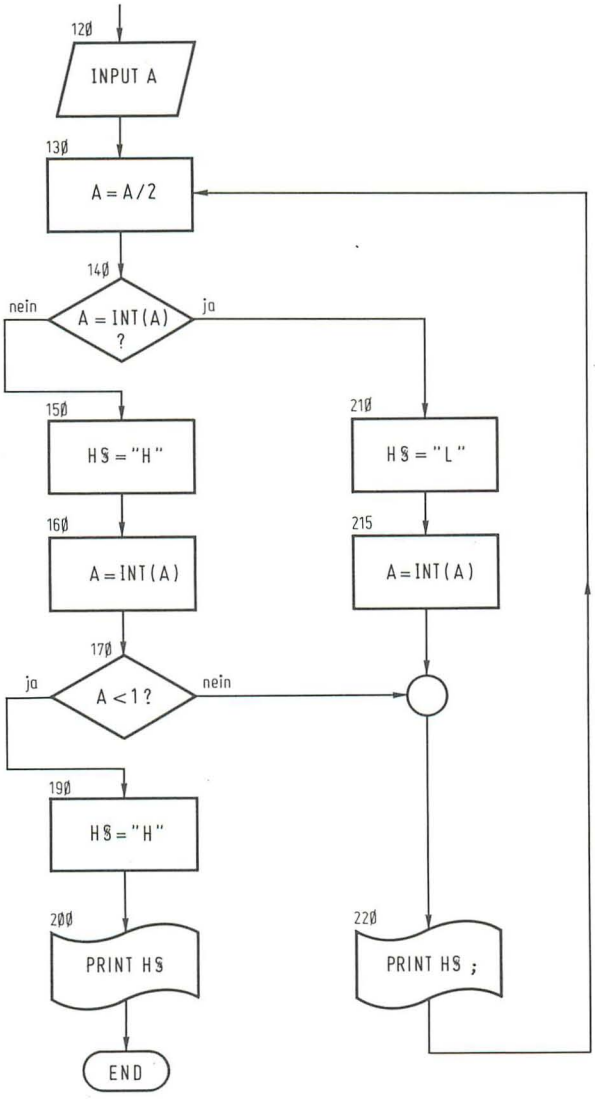


Abb. 11 Flußplan für ein Programm zur Umwandlung einer Dezimalzahl in eine Binärzahl

Nun lassen Sie Ihren Computer die Arbeit machen! Er soll:
- das Teilen durch 2 erledigen
- feststellen, ob ein Rest bleibt und daraus seine Schlüsse im Sinne der Abbildungen 9 und 10 ziehen!

Wir entwerfen dazu den Flußplan, *Abb. 11*. Wozu haben Sie das gelernt?

Den Entwurf aus Abb. 11 setzen Sie in folgendes Programm um:

```
  5 CLS
100 PRINT"BITTE GEBEN SIE DIE DEZIMALZAHL EIN!"
120 INPUT A
130 A=A/2
140 IF A = INT(A) THEN 210
150 H$="H"
160 A=INT(A)
170 IF A<1 THEN 190
180 GOTO 220
190 H$="H"
200 PRINT H$
205 END
210 H$="L"
215 A=INT(A)
220 PRINT H$;
230 GOTO 130
```

Dieses Programm macht genau das, was in den Abbildungen 9 und 10 aufgezeichnet war. Es hat nur noch einen „Schönheitsfehler"; Sie müssen das Ergebnis von rechts nach links lesen. Das wollen wir natürlich nicht. Wie läßt sich das verbessern?

Versuchen Sie, die gefundenen „H" und „L" als *indizierte* Stringvariable zu gewinnen! Dazu wird das Programm wie folgt ergänzt bzw. abgeändert (es sind nur diese Änderungen/Ergänzungen aufgeführt, alles andere bleibt wie oben):

```
  3 CLEAR 1000
  4 DIM N$(50)
 10 I = 0
140 IF A = INT(A) THEN 208
145 I = I + 1
150 N$(I)="H"
190 N$(I)="H"
200 PRINT N$(I)
208 I = I + 1
210 N$(I)="L"
220 PRINT N$(I);
```

Wenn Sie das so abgewandelte Programm starten, werden Sie das gleiche Ergebnis erhalten wie vorhin. Also, was soll's?

Sie haben zwar N$(I) als indizierte Größe erhalten, aber bei der Ausgabe davon keinen Gebrauch gemacht. Holen sie das nach, indem Sie noch einmal an dem obigen Programm ändern.

230 entfällt!

```
180 GOTO 130
200 PRINT N$(5);N$(4);N$(3);N$(2);N$(1)
220 GOTO 130
```

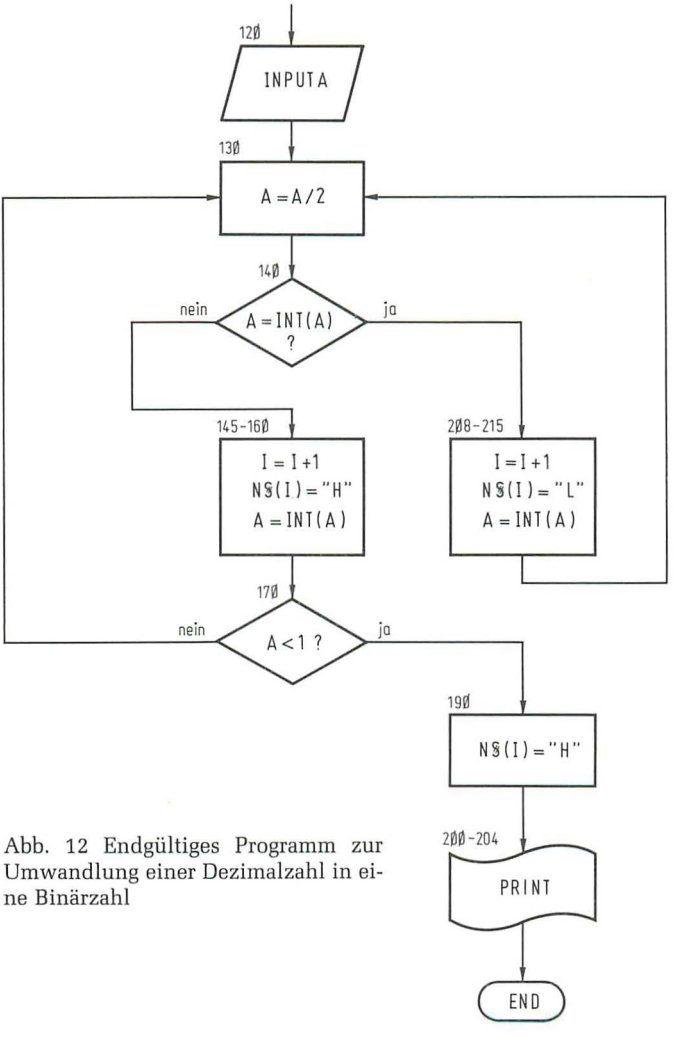


Abb. 12 Endgültiges Programm zur Umwandlung einer Dezimalzahl in eine Binärzahl

Weil wir in Zeile 2ØØ die I in N$(I) von 5 nach 1 schreiben lassen, erhalten wir tatsächlich für A = 28 den „Ausdruck" HHHLL.

Nun machen Sie das Programm noch „allgemeingültig"; d. h. für beliebige Stellenzahlen:

```
2ØØ FOR Z = I TO 1 STEP -1
2Ø2 PRINT N$(Z);
2Ø4 NEXT Z
2Ø5 END
```

Für Ihre Programmsammlung zum Schluß noch einen Flußplan (*Abb. 12*) vom gesamten Programm!

8.4 Sie verwandeln eine Binärzahl in eine Dezimalzahl

Das ganze Thema können Sie natürlich auch rückwärts abspulen lassen; Sie können eine gegebene Binärzahl in eine Dezimalzahl verwandeln. Welches Umwandlungsschema dafür gilt, ist in der *Abb. 13* dargestellt.
Es ist dort am Beispiel gezeigt, wie die Binärzahl HLLLL in die Dezimalzahl 16 verwandelt wird. In einem Computerprogramm – das dieselben Aufgaben erfüllen soll – müßte:

- die Länge der Binärzahl ermittelt und um 1 vermindert werden: A = LEN(A$)−1
- Es wird ein Zähler benötigt, der die Zeichenzahl des String A$ rückwärts bis Ø zählt.

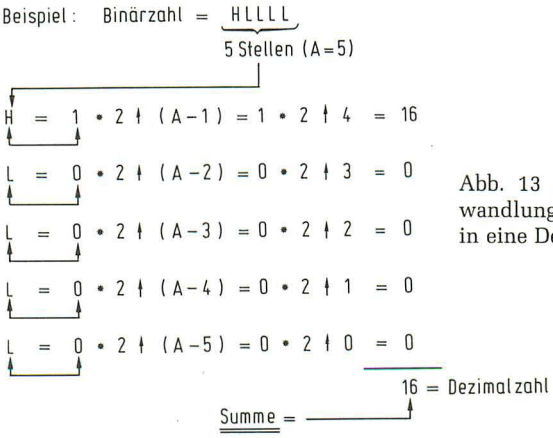


Abb. 13 Schema zur Umwandlung einer Binärzahl in eine Dezimalzahl

– Es werden gemäß dem Schema in Abb. 13 die Potenzen von 2 ausgerechnet.
– Es wird die Summe dieser Potenzen gebildet und als Ergebnis ausgegeben.
Das führt zunächst zu folgendem Programmentwurf:

1. Ausgangssituation

```
 5 CLS
10 INPUT"BINAERZAHL EINGEBEN !";A$
20 A= LEN(A$)-1 :Z = 1 : S = 0
```

2. Entscheidung, ob eine Potenz von 2 zu berechnen ist
Das ist nach dem Schema Abb. 13 dann der Fall, wenn das untersuchte Zeichen des String A$ ein „H" ist.

```
30 IF MID$(A$,Z,1)="H" THEN 70
```

Ist das NICHT der Fall, dann werden vor der Untersuchung des 2. Zeichens des String (Z=2):

– Z um 1 erhöht
– A um 1 vermindert

Das führt zu:

```
40 Z=Z+1 : A=A-1
```

Ist aber die Aussage in 30 *wahr*, dann muß:
– 2 zur A-ten Potenz erhoben werden
– die Summe der Potenz(en) gebildet werden.

Daneben ist – wie oben:
– Z um 1 zu erhöhen
– A um 1 zu vermindern
Das führt zu:

```
70 D = 2 ↑A : S = S + D : Z = Z +1 : A = A -1
```

3. Abbruch des Spieles
In beiden Zweigen von Zeile 30 muß das Spiel solange getrieben werden, bis alle Zeichen des String untersucht sind.

Das ist erreicht, wenn A < 0.

Das führt zunächst zu dem Programmschritt:

```
60 GOTO 30   bzw.
90 GOTO 30
```

sowie zu:

```
50 IF A < 0 THEN 500
80 IF A < 0 THEN 500
```

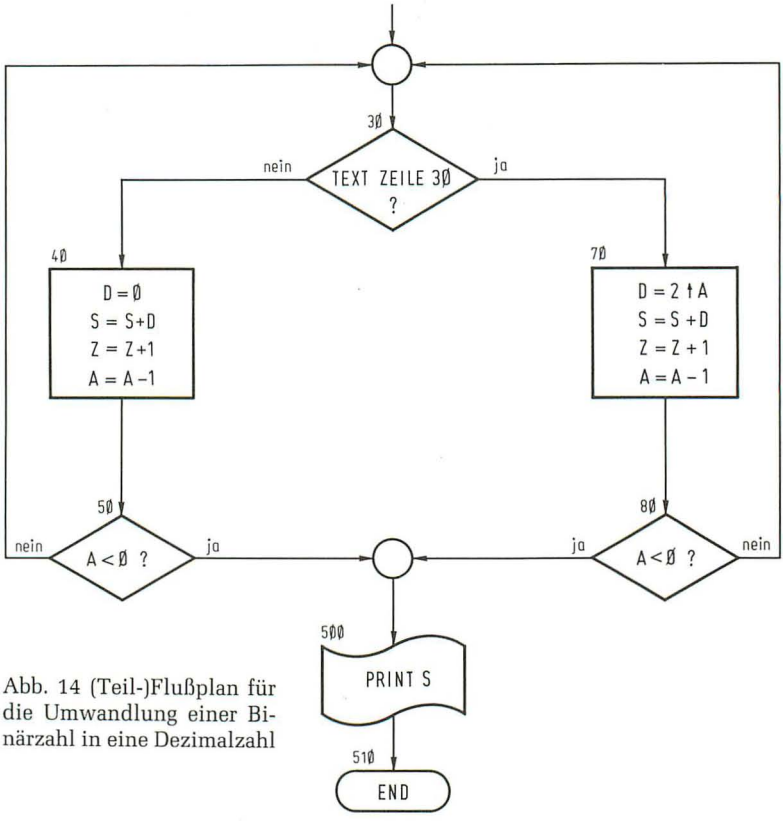


Abb. 14 (Teil-)Flußplan für die Umwandlung einer Binärzahl in eine Dezimalzahl

4. *Ausgabe des Ergebnisses*
Es bleibt noch zu tun:
5ØØ PRINT "DIE DEZIMALZAHL IST :";S
51Ø END
Ich denke, Sie waren so ehrgeizig, den Ablaufplan zu meiner Story bereits zu zeichnen? Dann vergleichen Sie den mit der *Abb. 14*!

8.5 Sie schreiben Geldbeträge in Worten

Wenn Sie einen Scheck über DM 35.– ausschreiben, dann müssen Sie diesen Betrag als Zahl in ein bestimmtes Feld eintragen.

An anderer Stelle des Schecks müssen Sie diesen Betrag in Worten wiederholen: ...fünfunddreißig.

Das folgende Programm soll Ihren Computer in die Lage setzen, dieses Wiederholen „in Worten" automatisch durchzuführen; Sie müssen dazu den jeweiligen Betrag nur als Zahl eingeben.

Neben einer weiteren Einübung von Stringfunktionen verfolgt dieses Programm den Zweck, Ihnen das Hantieren mit „DATA-Listen" und den geschickten Zugriff darauf nahezubringen.

Es versteht sich, daß jede Zahl, die der Computer in Worten ausgeben soll, vorgegeben und in irgendeiner Form im Programm enthalten sein muß. Wir wollen die Worte in einer DATA-Liste unterbringen und das Programm damit beginnen.

```
1000 DATA "EINS","ZWEI","DREI","VIER","FUENF","SECHS","SIEBEN"
1010 DATA "ACHT","NEUN","ZEHN","ELF","ZWOELF","DREIZEHN"
1020 DATA "VIERZEHN","FUENFZEHN","SECHSZEHN","SIEBZEHN"
1030 DATA "ACHTZEHN","NEUNZEHN","ZWANZIG"
```

So, das soll fürs erste ausreichen! Es geht weiter mit der Vorbereitung zum Programm:

```
5 CLS : CLEAR 1000
10 DIM U$(50):A$="UND":B$="DM":C$="DPF"
100 INPUT"BITTE BETRAG EINGEBEN !";N
```

Mit der folgenden Schleife werden die ab 1000 deponierten Daten in das Programm eingelesen:

```
500 FOR I = 1 TO 20
510 READ U$(I)
520 NEXT I
```

Beachten Sie: Obwohl wir in Zeile 10 U$ bis max. 50 DIMensioniert haben, dürfen wir in Zeile 500 nur bis 20 einlesen! ...Weil wir erst 20 Daten in unserer DATA-Liste haben!

Schreiben Sie nun den Programmteil für die Ausgabe!

```
700 CLS
710 I = N
720 PRINT N;B$;" SIND IN WORTEN: "
730 PRINT U$(I);B$
740 END
```

Machen Sie nun den ersten Testlauf; geben Sie dazu für N nur ganze Zahlen zwischen 1 und 20 ein.

Ich denke, es hat funktioniert. Verantwortlich dafür ist u. a. die Zeile 710, in der I zu N gemacht wurde. Geben Sie z. B. 17 ein, dann ist auch I siebzehn. Für U$(I) wird dann in Zeile 730 U$(17) ausgegeben. Und an 17. Stelle in der Datenliste steht: Siebzehn.

Sie brauchen jetzt nur noch die DATA-Liste mit EINUNDZWANZIG, ZWEIUNDZWANZIG...NEUNHUNDERTNEUNUNDNEUNZIGTAUSEND... fortzusetzen, und fertig wäre das Programm.

Fertig wär's zwar. Aber auch sehr umständlich und einfältig. Versuchen Sie einen anderen Weg; setzen Sie zunächst die Datenliste fort mit:

```
1040 DATA "DREISSIG","VIERZIG","FUENFZIG","SECHZIG","SIEBZIG"
1050 DATA "ACHTZIG", "NEUNZIG", "HUNDERT"
```

Ahnen Sie, worauf das hinausläuft? Wir wollen bei der Ausgabe beispielsweise der Zahl 35 diese aus den Elementen "DREISSIG" und "FUENF" der Datenliste zusammensetzen. Das zwingt uns zu einer Änderung der Zeile 710, in der nicht länger I = N sein kann.
Zwar ist noch U$(20) = ZWANZIG
aber U$(21) = DREISSIG und nicht EINUNDZWANZIG; dieser Text kommt in der Datenliste überhaupt nicht vor.

Wenn Sie die Zahl 35 sehen und „FUENF UND DREISSIG" sagen, dann hängen Sie doch den Text für den Zehnerwert an den für den Einerwert. Und das soll künftig der Computer tun! Dazu soll er zunächst 35 so zerlegen, daß er einen Wert Z für „ZEHNER" und einen E für „EINER" erhält. Den „Zehnerwert" gewinnt er aus: Z = INT(N/10)

Beispiel:
N = 35
N/10 = 3.5
INT(N/10) = 3
Damit ist Z (für Zehner) = 3

Den „Einerwert" gewinnt der Computer aus: E = N−Z*10

Beispiel:
N = 35
Z*10 = 30
N−Z*10 = 5
Damit ist E (für Einer) = 5

Was den Zugriff auf unsere Datenliste angeht, wären wir für die Einer aus dem Schneider. Denn: I = E ergibt – z. B. – I = 5. Und das fünfte „Datum" ist FUENF.

Bei den Zehnern wird das nicht klappen. Denn für Z = I = 3 erhalten wir das dritte „Datum" – und das ist DREI; nicht DREISSIG.

Vollziehen Sie einmal meine Überlegungen nach, die ich für die Zahl 30 angestellt habe: „DREISSIG" ist in der Datenliste auf Position 21...
...für 21 kann ich schreiben 20+(Z−2). (Erinnern Sie sich: für 30 ist Z = 3!)

Versuchen Sie, ob das auch für 40 und 50 paßt!

40 ist in der Datenliste auf Position 22. Da bei 40 Z den Wert 4 erreicht, gilt auch hier: 22 = 20+(Z−2).

Und auch bei 50 ergibt sich: 23 = 20+(Z−2).

Damit können wir ganz allgemein behaupten: Das zu jedem Zehner bis einschließlich EINHUNDERT gehörende Wort finde ich in der Datenliste auf *Platz I = 2Ø+(Z−2)*.

Den Beweis für diese Behauptung treten Sie an, indem Sie das Programm fortsetzen:

```
5ØØ FOR I = 1 TO 28      (Sie haben jetzt 28 Daten in der Liste!)
53Ø Z = INT(N/1Ø)
54Ø E = N−(Z*1Ø)
71Ø I = 2Ø+(Z−2)
```

Nun testen Sie Ihr Programm; geben Sie dazu nur ZEHNER zwischen 3Ø und 1ØØ ein. Sehen Sie, wie das arbeitet?

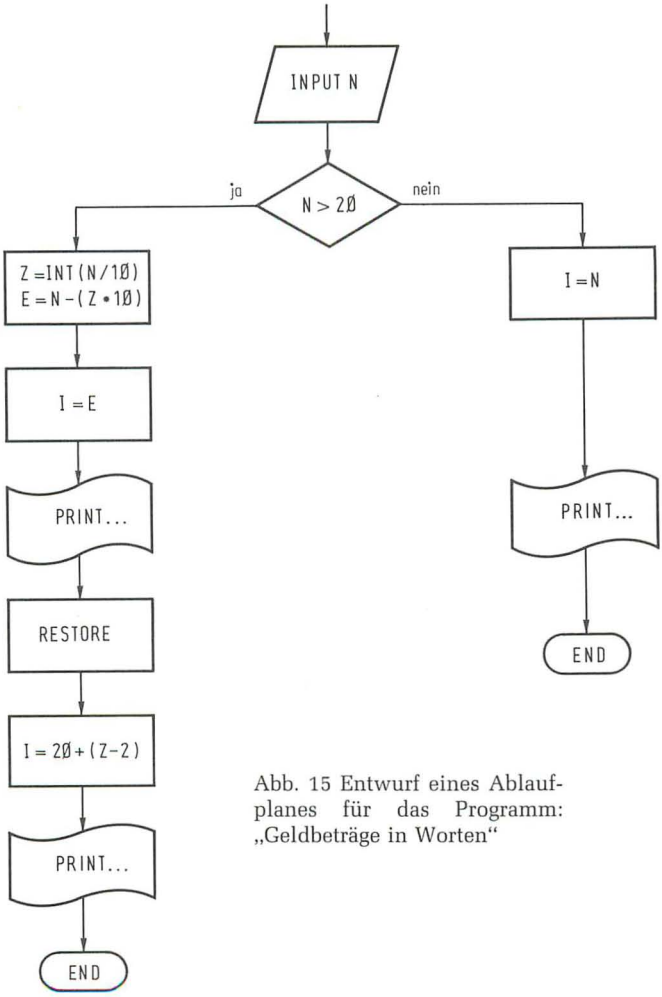


Abb. 15 Entwurf eines Ablaufplanes für das Programm: „Geldbeträge in Worten"

Bis hierher haben wir das Programm „geradeaus" entwickelt. Von nun an kommen Verzweigungen, und da sollte ein Flußplan her. Oder wenigstens ein Entwurf (*Abb. 15*).

Daraus ergibt sich folgender Programmaufbau:

```
550 IF N > 20 THEN 570
560 GO TO 700
570 Z = INT(N/10)
580 E = N-(Z*10)
590 I = E
595 CLS
600 PRINT N;B$;" SIND IN WORTEN :"
610 PRINT U$(I);
620 RESTORE
630 I = 20+(Z-2)
640 PRINT A$+U$(I)+B$
650 END
```

Löschen Sie 530, 540 und ändern Sie 710 in: I=N

Das Einzige, was Sie sicher ebenso wie mich stört, ist die Aussage: ...UNDEINHUNDERT... für N = 100.

Das können Sie sicher selbst beseitigen, wenn Sie nach dem bisherigen Muster das Programm für Zahlen über 100 weiterentwickeln – wozu ich Sie animieren möchte.

Zwei Hinweise schulde ich Ihnen noch:

- Falls Sie sich über 620 RESTORE wundern: Sie benutzen ja ab 620 die „alte" Datenliste für eine „neue" Operation. Und dazu schreibt BASIC vorher RESTORE vor!
- In Zeile 640 steht: 640 PRINT A$+U$(I)+B$
  Die Pluszeichen betreiben hier die „Concatenation"; diesen Begriff habe ich Ihnen an anderer Stelle schon erklärt.

Für die Weiterentwicklung des Programms für Beträge über 100 DM hinaus können Sie das bisherige als Anregung verwenden und sich selbst „testen".

Ich möchte hier mit Ihnen zusammen das Programm in die Lage versetzen, auch mit Bruchteilen von DM fertig zu werden. Sie sollten auch Beträge eingeben können wie 1.35, 0.90 oder 87.20. Dabei sollten Sie sich erinnern, daß man auf einem Scheck üblicherweise nur die DM-Beträge in Worten, die Pfennige aber in Zahlen schreibt. Ob in Ihrer Eingabe für N Pfennige vorkommen, stellen Sie wie folgt fest:

Ist INT(N) *kleiner als N*?

Ist das der Fall, dann enthält Ihre Eingabe Pfennige; ist das NICHT der Fall, dann enthält N keine Pfennige.

Beispiel: N = 9Ø
dann ist INT(N) *gleich* N
N = 9Ø.55
dann ist INT(N) = 9Ø und damit *kleiner als N.*

Wie viele Pfennige P in N enthalten sind, erfahren Sie durch:
P = N – INT(N)

Beispiel: N = 9Ø
P = 9Ø – 9Ø = Ø; es sind keine Pfennige in N enthalten.
N = 9Ø.55
P = 9Ø.55 – 9Ø = .55; es sind 55 Pfennige in N enthalten.

Das hat noch einen Schönheitsfehler! Im zweiten Beispiel ergab sich für P = .55. Herausbekommen wollen Sie aber 55 Pfennige!

Multiplizieren Sie P mit 1ØØ!

Also: P = (N–INT(N))*1ØØ : P=INT(P*1ØØ+.5)/1ØØ

(Mit dem Runden wollen wir uns vor Überraschungen durch die Rechenkünste des Computers schützen!)

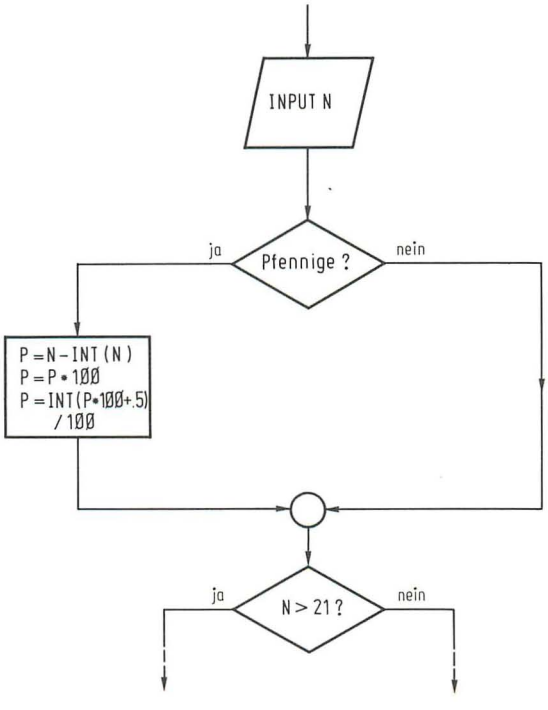


Abb. 16 Mit diesem „Filter" prüfen Sie, ob Ihre Eingabe für N „Pfennige" enthält

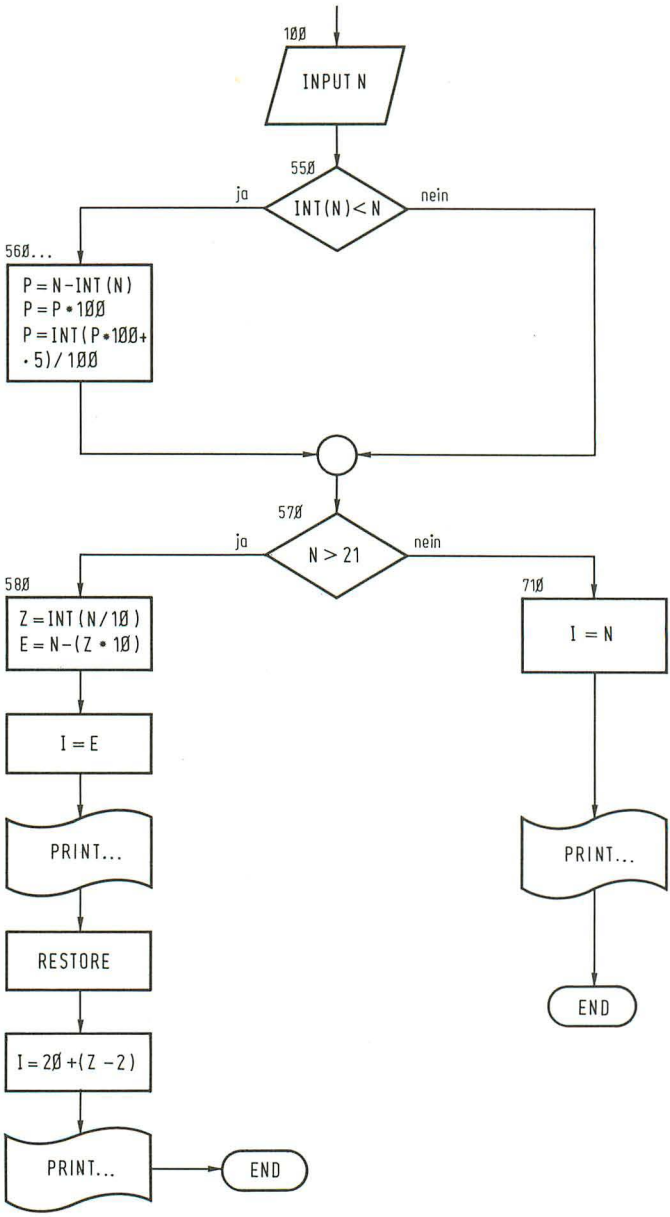


Abb. 17 Endgültiger Ablaufplan

Um nun in dem Computerprogramm auch die Pfennige berücksichtigen zu können, müssen wir dem „Filter" in Zeile 55Ø ein weiteres vorschalten (*Abb. 16*). Ihr endgültiges Programm erhält dadurch folgenden Wortlaut:

```
5 CLEAR 1ØØØ:DIM U$(5Ø) : CLS : A$="UND" : B$="DM" : C$="DPF"
1ØØ INPUT"BITTE BETRAG EINGEBEN !";N
5ØØ FOR I = 1 TO 28
51Ø READ U$(I)
52Ø NEXT I
55Ø IF INT(N) < N THEN 56Ø
555 GOTO 57Ø
56Ø P = (N-INT(N))*1ØØ
565 P = INT(P*1ØØ+.5)/1ØØ
57Ø IF N > 21 THEN 58Ø
575 GOTO 7ØØ
58Ø Z = INT(N/1Ø)
585 E = N-(Z*1Ø)
59Ø I = E : CLS
6ØØ PRINT N;B$;" SIND IN WORTEN :":PRINT
61Ø PRINT U$(I);
62Ø RESTORE
63Ø I = 2Ø+(Z-2)
64Ø PRINT A$+U$(I)+B$+"        ";P;C$
65Ø END
7ØØ CLS
71Ø I = N
72Ø PRINT N;B$;" SIND IN WORTEN " : PRINT
73Ø PRINT U$(I)+B$+"        ";P;C$
74Ø END
```

Zum Schluß:
Übertragen Sie – falls noch notwendig – die DATA-Liste mit den Programmzeilen 1ØØØ...1Ø5Ø.
Nun folgt noch der endgültige Ablaufplan (*Abb. 17*):

8.6 Sie entwerfen ein Wörterbuch

Oder sagen wir besser: Sie entwerfen ein Konzept dazu. Grundlage des Konzeptes ist eine geschickte Anordnung von Daten in einer DATA-Liste und ein lösungsgerechter Zugriff darauf.

Ich zeige Ihnen in der *Abb. 18*, wie ich mir die Datenanordnung für ein Wörterbuch – das Deutsch, Englisch und Französisch beherrschen soll – vorstelle. Wenn ich mich dabei mit wenigen Daten begnüge, bitte ich um Ihre Nachsicht.

Sprache (Index : B)

Worte (Index: A)

DEUTSCH	ENGLISCH	FRANZÖSISCH
fünf	five	cinq
sechs	six	six
sieben	seven	sept
acht	eight	'huit

Abb. 18 Aufbau der Datenliste für ein Lexikon

Nicht das *Programm* für ein Wörterbuch ist das Ziel. Sie sollten vielmehr erkennen, welcher Grundgedanke diesem Programm zugrundeliegt. Sie können dann nämlich selbst und leicht Abwandlungen programmieren: Tabellenbücher, Erklärungen für amerikanische „Computer-Fremdworte" in deutscher Sprache usw. Sehen Sie sich Abb. 18 an!

Sie erkennen, daß ich ein *zweidimensionales* (Daten-)Feld angelegt habe. In horizontaler Richtung B erstrecken sich die Sprachen. Es sind im Beispiel drei; Sie können sich leicht vorstellen, daß man das auch auf N treiben kann (die Grenze bildet nur die Speicherkapazität Ihres Computers). In vertikaler Richtung A erstreckt sich der Wortschatz; sehen Sie mir nach, wenn ich mich mit vier Worten begnüge.

Das Datenfeld wird angelegt mit:

```
10 CLEAR 1000 : DIM A$(4,3)
```

Darin ist A$ das jeweilige Wort. In ( ) steht der Platz des Wortes in der Datenliste; hier bedeutet ...(4,3), daß es 4 Zeilen und 3 Spalten gibt. Legen wir zunächst die Datenliste an!

```
70 DATA "FUENF","FIVE","CINQ"
80 DATA "SECHS","SIX","SIX"
90 DATA "SIEBEN","SEVEN","SEPT"
100 DATA "ACHT","EIGHT","HUIT"
```

Als nächstes kommt es darauf an, diese Liste so in den Computer einzulesen, daß sie darin genau so steht, wie in der Abb. 18.

```
20 FOR A = 1 TO 4
25 FOR B = 1 TO 3
30 READ A$(A,B)
40 NEXT B
50 NEXT A
```

Es folgt der Teil des Programmes, der die Daten in der gewünschten Form ausgibt. Dazu fragen Sie den Benutzer, in welche Sprache das deutsche Wort übersetzt werden soll!

```
200 PRINT" IN WELCHE SPRACHE SOLL UEBERSETZT WERDEN ?"
205 PRINT" GEBEN SIE FUER          ENGLISCH ..............  2 EIN !"
207 PRINT" GEBEN SIE FUER          FRANZOESISCH...........  3 EIN !"
210 INPUT B
```

Wir schenken uns hier die Sperren gegen Falscheingabe; wenn Sie das Programm nutzen wollen, sind Sie inzwischen Mann's genug, diese selbst einzufügen. Beschäftigen wir uns lieber mit der Ausgabe!

```
220 INPUT" GEBEN SIE DAS DEUTSCHE WORT EIN !";X$
225 FOR A = 1 TO 4
230 IF A$(A,1)=X$ THEN PRINT A$(A,B)
240 NEXT A
```

Fertig. So einfach ist das!

9 Sie editieren

9.1 Was bedeutet EDIT?

In einem Englisch/Deutschen Wörterbuch finden Sie unter:
EDITOR = Herausgeber, Schriftleiter; und für das „Tätigkeitswort"
To edit = herausgeben, redigieren. Und was hat das mit BASIC zu tun?

Nun, in manchen Betriebssystemen von Homecomputern gibt es einen EDITOR, das ist eine Routine, die es erlaubt, von Ihnen erstellte BASIC-*Programme* zu editieren, zu bearbeiten.

nicht beim C64

9.2 Was beherrschen Sie schon?

Nun dürfte es für Sie nichts Neues sein, Programme zu bearbeiten; zu „editieren" – auch wenn Sie das Wort EDIT noch nie gehört haben. Bevor wir uns mit dem EDITieren beschäftigen, wollen wir kurz zusammenfassen, was Sie schon editieren können.

1. Beispiel
Sie wollen die Programmzeile schreiben:
10 PRINT"UEBUNG"
Sie bemerken bei
10 PRINT"UBU
daß Sie das E als zweiten Buchstaben vergessen haben.

Sie bessern bekanntlich Ihren Fehler aus, indem Sie durch Druck auf die Cursor-Steuertaste um zwei Positionen nach links gehen, auf die Taste E drükken... und der Schaden ist behoben.

Sie haben hier editiert – wenn auch in einer noch nicht mit RETURN oder ENTER abgeschlossenen Zeile.

2. Beispiel
Sie wollen die bereits vom Computer übernommene Programmzeile:
10 PRINT A*7
aus dem Programm wieder verschwinden lassen.
Sie drücken einfach 10, danach ENTER oder RETURN. Fertig.

Wollen Sie die Programmzeile 1Ø nur ändern, dann schreiben Sie halt:
1Ø PRINT A*6

Sie haben die Zeile überschrieben; den alten Inhalt durch den neuen ausgetauscht.

Der langen Rede kurzer Sinn: Sie können Ihre BASIC-Programme auch ohne EDITieren recht ordentlich bearbeiten. Und da ohnehin nicht jede Maschine über einen (Programm-)EDITor verfügt, will ich Ihnen nur an ein paar Beispielen zeigen, wie man damit in BASIC-Programmen hantiert, um auszubessern oder zu ändern.

9.3 Einige Beispiele

Für die Beispiele, mit denen ich Ihnen EDIT vorführen will, verwenden wir folgendes Übungsprogramm:

```
 5 CLS
1Ø PRINT"UEBUNGS-PROGRAMM"
2Ø A=1
3Ø PRINT A
4Ø A=A+1:IF A=1Ø THEN 6Ø
5Ø GOTO 3Ø
6Ø PRINT"A=1Ø":END
```

Mit den folgenden fünf Beispielen wollen wir dieses Programm mit Hilfe von EDIT in der einen oder anderen Zeile verändern. Beurteilen Sie die Nützlichkeit – die ja in der bequemen und zeitsparenden Bearbeitung von Programmen liegen müßte – selbst! Übrigens: die bei EDIT verwendeten Kürzel wie H, N usw. sind NICHT Bestandteil der Programmiersprache BASIC!

1. EDIT-Kommando: EDIT (Zeilennummer)

Erklärung:
Dieses Kommando führt in den EDIT-Bereich. Es muß angegeben werden, welche (Programm-)Zeile editiert werden soll.
Tasten-Manipulation(en)
EDIT 1Ø

[ENTER]

Auswirkungen auf Übungsprogramm:
Der Schirm zeigt:
1Ø_

Jeder Druck auf die *Leertaste* läßt den Cursor um eine Position nach rechts rücken; gleichzeitig erscheint auf dem Schirm das betreffende Zeichen. 5× *drücken der Leertaste* ergibt:

1Ø PRINT_

Das gleiche Ergebnis erhält man:

Taste [% 5], Leertaste (nur einmal!)

Weitere Erläuterungen:

- Ausstieg aus dem EDIT-Bereich: Druck auf:

  [ENTER] . Dabei werden etwa getroffene Änderungen in der EDITierten Zeile berücksichtigt.
- Auch bei EDITieren gilt ein „Blank" als Zeichen!
- Die beim EDITieren eingegebenen Zeichen (oben: 5!) werden NICHT angezeigt.

2. EDIT-Kommando: N, [←]

Erklärung:

- Jedes Betätigen der [←]-Taste bringt den Cursor um eine Position nach links.
- [% 5] [←] bringt den Cursor 5 Positionen nach links usw.

Tasten-Manipulation(en)

a) [ENTER]

b) EDIT 4Ø
[ENTER]

c) [% 5] Leertaste

d) [% 5] [←]

Auswirkungen auf Übungsprogramm:

a) Ausstieg aus dem EDIT-Bereich
b) Einstieg in EDIT, Zeile 4Ø
Bildschirm zeigt: 4Ø_
c) 5 Positionen nach rechts
Bildschirm zeigt: 4Ø A = A + 1_
d) 5 Positionen nach links
Bildschirm zeigt: 4Ø_

Weitere Erläuterungen:
Bei d) verschwinden die (5) Zeichen zwar wieder vom Bildschirm, werden aber NICHT aus der Programmzeile gelöscht!

3. EDIT-Kommando: X

Erklärung:
– Die aktuelle Zeile wird angezeigt
– Der Cursor geht an das Ende der Zeile
– Durch entsprechende Eingabe können *weitere Befehle angehängt werden!*

Tasten-Manipulation(en)

a) [ENTER]

b) EDIT 3Ø

[ENTER]

c) X
d) , A * 2

[ENTER]

Auswirkungen auf Übungsprogramm:
a) Ausstieg aus dem EDIT-Bereich
b) Einstieg in EDIT, Zeile 3Ø
Bildschirm zeigt: 3Ø_
c) X
Bildschirm zeigt: 3Ø PRINT A_
d) nach Eingabe von ,A * 2; Ausstieg aus EDIT durch [ENTER]

zeigt Bildschirm geänderte Zeile wie folgt: 3Ø PRINT A, A * 2

Weitere Erläuterungen:

- Nach Eingabe von X *NICHT* [ENTER] drücken!
- Es ist auch folgende Korrekturmöglichkeit gegeben:
  Nach Eingabe von X erscheint 3Ø PRINT A_
- Ein Druck auf die Taste [←] ergibt: 3Ø PRINT_
- Korrektur: (Beispiel!) A * 2, A * 3
- *Neue Zeile 3Ø* lautet nun: 3Ø PRINT A * 2, A * 3
- Also: ist X gedrückt, wird durch jeden Druck auf [←] das entsprechende Zeichen aus der Programmzeile *gelöscht!*

4. EDIT-Kommando: I

Erklärung:
– erlaubt Einsetzen (Insert) neuer Befehle ab einer gewählten Cursorposition.
Tasten-Manipulation(en)

a) [ENTER]

b) EDIT 4Ø

[ENTER]

c) Leertaste so oft drücken, bis auf dem Schirm erscheint: 4Ø A = A + 1_
d) I drücken!
e) .5 eingeben!

f) [ENTER]

Auswirkungen auf Übungsprogramm:
a) Ausstieg aus dem EDIT-Bereich
b) Einstieg in EDIT, Zeile 4Ø
c) Auf den Bildschirm bringen: 4Ø A = A + 1_ (durch Leertaste!)
d) I drücken!
e) .5 eingeben! Bildschirm zeigt: 4Ø A = A + 1.5_

f) Nach [ENTER] heißt die geänderte Zeile:

4Ø A = A + 1.5 : IF A = 1Ø THEN 6Ø

Weitere Erläuterungen:
- Nach Eingabe von I NICHT [ENTER] drücken!
- Nach Eingabe von I wird durch jeden Druck auf [←] das entsprechende Zeichen aus der Programmzeile *gelöscht!*

5. EDIT-Kommando: H

Erklärung:
„AbHacken" des Zeilenrestes ab einer bestimmten Cursorposition und Einsetzen eines neuen Befehls.
Tasten-Manipulation(en)

a) [ENTER]

b) EDIT 4Ø

[ENTER]

c) Leertaste so oft drücken, bis auf dem Schirm erscheint:
40 A = A + 1,5 : IF A = 1_
d) H drücken!
e) 5 THEN 60 eingeben!
f) [ENTER]

Auswirkungen auf Übungsprogramm:
a) Ausstieg aus dem EDIT-Bereich
b) Einstieg in EDIT, Zeile 40
c) Auf den Bildschirm bringen: 40 A = A + 1,5 : IF A = 1_
d) H drücken!
e) 5 THEN 60 eingeben!
f) Nach [ENTER] heißt die neue Zeile:

40 A = A + 1,5 : IF A = 15 THEN 60

Weitere Erläuterungen:
- Nach Eingabe von H *NICHT* [ENTER] drücken!
- Nach Eingabe von H wird durch jeden Druck auf [←] das entsprechende Zeichen aus der Programmzeile *gelöscht!*

10 Der Computer als Zeichenmaschine

Im ersten Teil dieses Abschnittes soll das „Zeichnen" nicht allzu wörtlich aufgefaßt; es soll als eine Sonderform des „Schreibens" betrieben werden.

Wir verwenden deshalb das Kommando PRINT und seine Variationen.

10.1 Zeichnen mit PRINT TAB...

Geben Sie folgendes Programm ein:

```
5 CLS
1Ø FOR A = 1 TO 4 STEP .2
2Ø B = A↑3
3Ø PRINT TAB (B) "*"
4Ø NEXT A
1ØØ GOTO 1ØØ
```

Das Ergebnis wird Sie vielleicht verblüffen. Sie können daraus folgende Merkregeln ableiten:

- PRINT TAB gilt bekanntlich nur für eine Zeile; Sie können den gewünschten Ausdruck auf jeder der Ø...63 Positionen einer Zeile erhalten.

- Als Argument haben Sie hier eine *Funktion* gewählt; nämlich B = A↑3. Sie hätten in Zeile 3Ø auch schreiben können: PRINT TAB (A↑3)

- Was Sie ausdrucken lassen, ist Ihrem Computer offensichtlich egal. Hier hat er auf jeder Zeile, immer auf der aktuellen Position B = A ↑ 3, einfach ein Sternchen gedruckt.

- Fazit: Sie können also mit PRINT TAB den Verlauf von mathematischen Funktionen „zeichnen".

- Die Endlosschleife in Zeile 1ØØ soll verhindern, daß die READY-Meldung des Computers unser Gemälde beeinträchtigt. Sie können deshalb ein derartiges Programm nur mit BREAK wieder verlassen.

Ein weiteres Beispiel:

```
5 CLS
10 FOR X = 0 TO 360 STEP 30
20 B = X*3.1416/180
30 Y = SIN(B)*10
40 PRINT TAB(20 + Y)"*"
50 NEXT
100 GOTO 100
```

Das Ergebnis wird eine Sinus-Kurve sein. Ich muß Ihnen noch erläutern:

Zeile 20: Hier wird der Winkel X ins Bogenmaß umgerechnet.

Zeile 30: Da alle Sinuswerte zwischen 0 und 1 liegen – was kein berauschendes Bild ergeben würde – haben wir jeden Sinuswert mit 10 multipliziert. Das wirkt wie eine „Lupe". Versuchen Sie doch einmal:
30 Y = SIN(B)

Zeile 40: Eine Sinuskurve hat einen „positiven" und einen „negativen" Kurvenzug. Damit der negative uns nicht nach links aus dem Bild fällt, haben wir durch ...(20 + Y)... das Bild um 20 Buchstabenpositionen nach rechts verschoben. Versuchen Sie einmal:
40 PRINT TAB (Y)"*"
40 PRINT TAB (30 + Y)"*"
Alles klar?

Zum Studium der Wirkung der beiden Konstanten 10 in Zeile 30 bzw. 20 in Zeile 40 soll folgendes Programm dienen:

```
5 CLS
7 INPUT"HOEHE DER AMPLITUDE =";A
9 INPUT"X-ACHSE BEI :";C
10 FOR X = 0 TO 360 STEP 30
20 B = X*3.1416/180
30 Y = SIN (B)*A
40 PRINT TAB (Y + C)"*"
50 NEXT
100 GOTO 100
```

Wenn Sie das ein paarmal laufen lassen, sind wohl sämtliche Unklarheiten beseitigt?

Einen Schönheitsfehler haben unsere Bildchen. Sie sind gewohnt, derartige Kurven mit horizontaler X-Achse zu zeichnen; der Computer zeichnet in unseren Beispielen mit vertikaler X-Achse.

Sie werden später erfahren, was man dagegen tun kann!

Versuchen Sie einmal, mit PRINT TAB eine einfache Figur zu zeichnen! Ich schlage Ihnen eine ganz besonders einfache vor; Sie können dann selbst Ihre Phantasie einsetzen!

```
  5 CLS
 1Ø PRINT:PRINT:PRINT
 4Ø PRINT TAB(31)"XXXX"
 5Ø PRINT TAB(3Ø)"X     X"
 6Ø PRINT TAB(29)"X       X"
 7Ø PRINT TAB(29)"X       X"
 8Ø PRINT TAB(29)"X       X"
 9Ø PRINT TAB(3Ø)"X     X"
1ØØ PRINT TAB(31)"XXXX"
2ØØ GOTO 2ØØ
```

Ob Ihnen das Ergebnis nun gefällt oder nicht:

- Das „Zeichnen" auf die hier beschriebene Art und Weise ist ein mühsames Unternehmen! Selbst für bescheidene Figuren ist ein großer Programmieraufwand nötig.
- Wegen der relativ großen Buchstaben (Positionen) ergibt sich ein recht grobes Raster; feine Konturen lassen sich so nicht darstellen.

10.2 Zeichnen mit PRINT

Der Vollständigkeit halber möchte ich Ihnen zeigen, daß derartige Zeichnungen auch mit dem einfachen Kommando „PRINT" möglich sind.

```
  5 CLS
 1Ø PRINT:PRINT:PRINT
 4Ø PRINT"                    XXXX"
 5Ø PRINT"                  X      X"
 6Ø PRINT"                X          X"
 7Ø PRINT"                X          X"
 8Ø PRINT"                X          X"
 9Ø PRINT"                  X      X"
1ØØ PRINT"                    XXXX"
2ØØ GOTO 2ØØ
```

In diesem Programm wurde also anstatt PRINT TAB(3Ø) einfach geschrieben PRINT – und dann 3Ømal die Leertaste gedrückt!

10.3 Zeichnen mit Strings

In dem nächsten Programm machen Sie zum Zeichnen von Strings Gebrauch, indem Sie Linienstücke oder „Bildelemente" in Form von Strings definieren und diese dann in der gewünschten Anordnung „Schreiben" bzw. „Zeichnen" lassen.

```
5 CLS
10 A$ ="*****************"
20 B$ ="*                 *"
30 PRINT A$
40 FOR X = 1 TO 10
50 PRINT B$
60 NEXT X
70 PRINT A$
100 GOTO 100
```

Das ist wohl nicht ganz so mühsam wie das letzte Beispiel?

10.4 Eine neue Zeichenhilfe: PRINT@

nicht bei C64

Sofern Ihr Computer über diese Erweiterung des PRINT-Kommandos verfügt, können Sie auch damit in bescheidenem Umfang zeichnen.

- Mit PRINT TAB erreichen Sie jede der 0...63 Positionen *einer Zeile.*
- Mit PRINT@ erreichen Sie jede der 16 mal 64 Positionen des *gesamten Bildschirms.*

Damit Sie mit diesem Kommando umgehen können, ist es zweckmäßig, die „Adressen" der 1024 Bildpositionen zu kennen. Die *Abb. 19* zeigt die Positionen auf dem Bildschirm, die *Tabelle 5* die zugehörigen „Adressen".

Machen Sie ein paar Übungen!

```
5 CLS
10 PRINT @ 0,"X"
20 PRINT @ 63,"X"
30 PRINT @ 320,"ANFANG"
40 PRINT @ 542,"MITTE"
50 PRINT @ 891,"ENDE"
60 PRINT @ 510,"FALSCH POSITIONIERT"
```

Ich denke, das reicht aus!

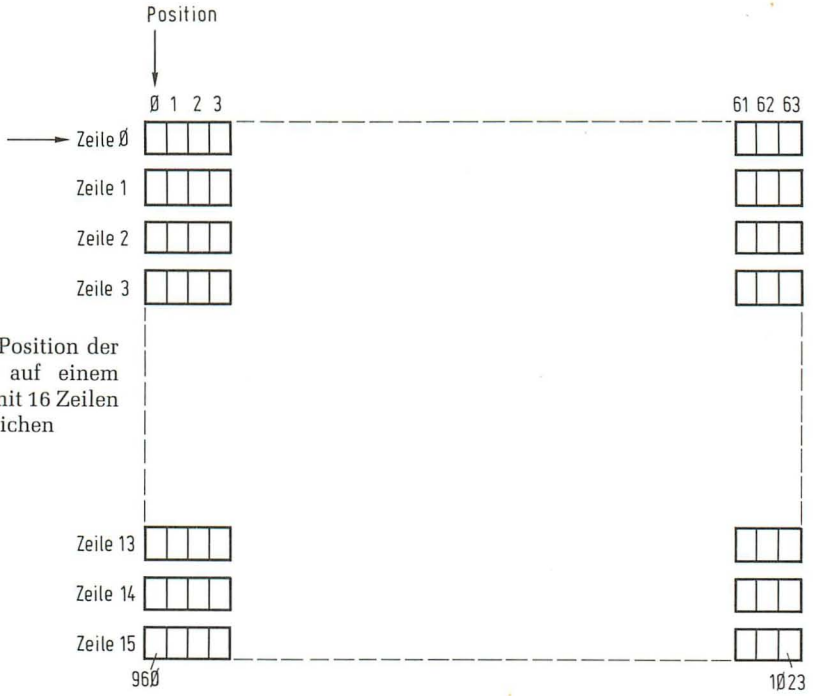


Abb. 19 Position der Zeichen auf einem Schirm mit 16 Zeilen zu 64 Zeichen

Tabelle 5: „Adressen" der Zeichenpositionen aus Abb. 19

ZEILEN	POSITIONEN		
Ø	Ø	→	63
1	64	→	127
2	128	→	191
3	192	→	255
4	256	→	319
5	32Ø	→	383
6	384	→	447
7	448	→	511
8	512	→	575
9	576	→	639
1Ø	64Ø	→	7Ø3
11	7Ø4	→	767
12	768	→	831
13	832	→	895
14	896	→	959
15	96Ø	→	1Ø23

Insgesamt: 1Ø24 Zeichenpositionen

10.5 SET, RESET

nicht C64

Die meisten Personal-Computer bieten die Möglichkeit, mit maschinentypischen Kommandos und Statements in mehr oder weniger bescheidenem Umfang grafische Darstellungen auf den Bildschirm zu zaubern.

In diesem Kapitel sind diese Möglichkeiten am Beispiel des TRS-80 erläutert. Für den Grafikbetrieb werden bei diesem Gerät wesentlich mehr „Bildpunkte" angesprochen als die insgesamt 16 mal 64 PRINT-Positionen.

- In der Horizontalen (X-Richtung) stehen 64 mal 2 = 128 Bildpunkte zur Verfügung.
- In der Vertikalen (Y-Richtung) stehen 16 mal 3 = 48 Bildpunkte zur Verfügung.
- Damit entfallen insgesamt 128 mal 48 = 6144 Bildpunkte auf den Bildschirm; jeder dieser Bildpunkte ist einzeln „ansprechbar".

Wie Zeichen- und PRINT-Positionen zusammengehören, zeigt die *Abb. 20*. Für „Zeichenarbeiten" ist eine derartige Übersicht von großem Nutzen.

- Gesetzt, „eingeschaltet" wird jeder Bildpunkt mit SET
- Zurückgesetzt, „ausgeschaltet" wird jeder Bildpunkt mit RESET
- Welcher Bildpunkt gemeint ist, wird jeweils in ( ) angegeben
- Der korrekte Ausdruck für das Setzen des Bildpunktes 1Ø,5 – das heißt:
  10. Bildpunkt auf der Horizontalen
  5. Bildpunkt auf der Vertikalen
  lautet demnach: SET(1Ø,5)
- Sinngemäß heißt es für das „Ausschalten" des gleichen Bildpunktes:
  RESET(1Ø,5)

Mit den folgenden Übungen werden Sie mit SET und RESET sehr schnell vertraut; wir begnügen uns fürs erste mit Manipulationen an einem einzelnen Bildpunkt.

```
5 CLS
1ØØØ GO TO 1ØØØ
```

Erster Schritt:

```
1Ø INPUT"WERT FUER X EINGEBEN !";X
2Ø INPUT"WERT FUER Y EINGEBEN !";Y
25 CLS
3Ø SET(X,Y)
```

Spielen Sie das einige Male; lassen Sie aber unser Mini-Programm stehen!
Fahren Sie fort:

```
6Ø RESET(X,Y)
```

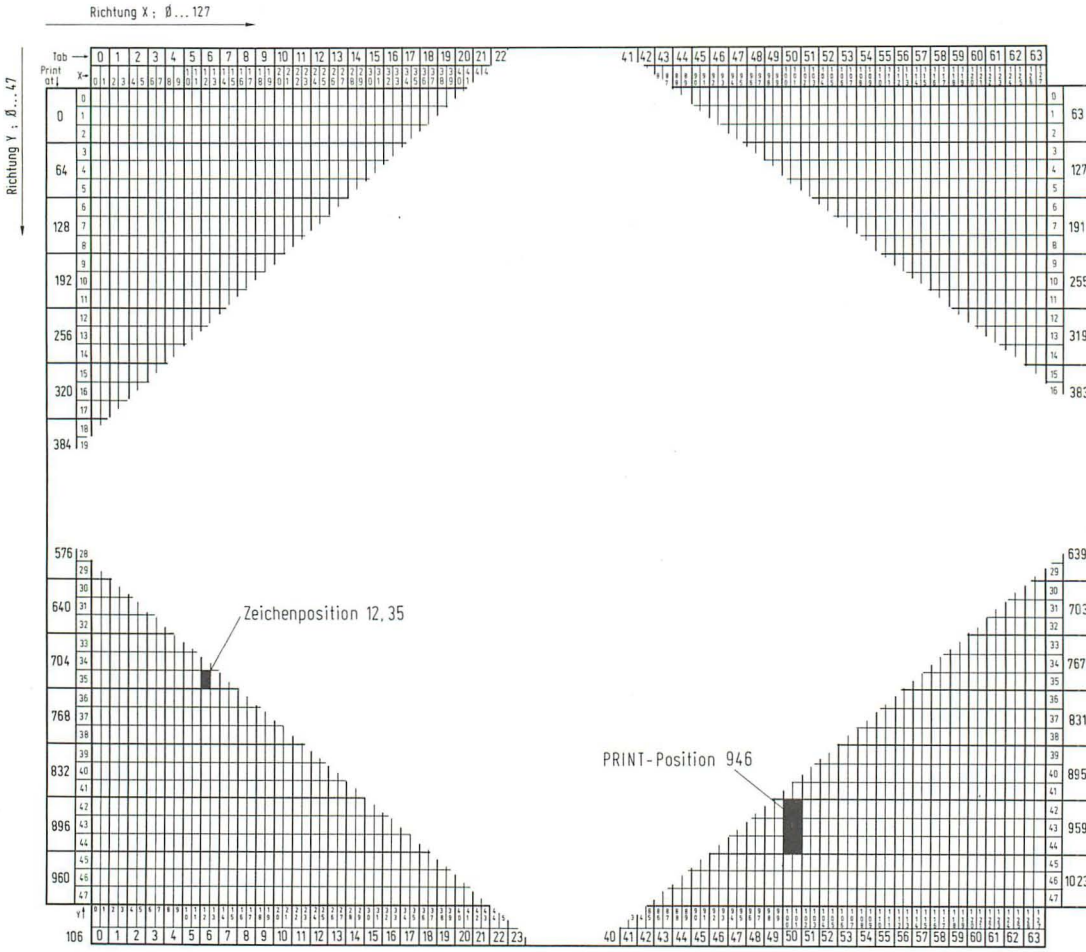


Abb. 20 Ausschnitt; Übersicht über die PRINT!!,PRINT TAB.. und PRINT a – Positionen des TRS 80

Wenn Sie überhaupt etwas sehen, dann sehen Sie den gewählten Lichtpunkt nur kurz aufblitzen; in 3Ø hatten Sie ihn ein-, in 6Ø sofort wieder ausgeschaltet.

4Ø FOR Z = 1 TO 1ØØØ:NEXT

Variieren Sie die Wartenschleife mit 1ØØ, 1Ø, 2ØØØ!
8Ø FOR Z = 1 TO 1ØØ
9Ø GO TO 3Ø
Sie haben soeben das Blinklicht erfunden!

Fügen Sie ein:
85 X = X + 1

Der Leuchtpunkt wandert horizontal von links nach rechts! Ersetzen Sie 85 durch:

85 Y = Y + 1

Der Leuchtpunkt wandert vertikal von oben nach unten! Am Ende erscheint jeweils ?FC ERROR IN 3Ø. Was bedeutet das?

Nun, sobald X größer als 127 bzw. Y größer als 47 wird, will der Bildpunkt das festgelegte Raster verlassen. Und das geht nicht an!

Bleiben Sie bei dem letzten Beispiel und ergänzen Sie:

86 IF Y > 47 THEN Y = Y − 48

Sinngemäß gilt:

85 X = X + 1
86 IF X > 127 THEN X = X − 128

Damit haben Sie dieses Problem gelöst!

Und nun kombinieren Sie wie folgt:

85 X = X + 1: Y = Y + 1
86 IF X > 127 THEN X = X − 128
87 IF Y > 47 THEN Y = Y − 48

Hier erreicht Y zuerst die Grenze des Bildschirms und wird auf Null zurückgesetzt; X hat zu diesem Zeitpunkt ebenfalls den Wert 47. Sie erhalten deshalb verschiedene, schräg von oben nach unten verlaufende Linien. Jede wird X + 47 Positionen weiter rechts angesetzt.

Schreiben Sie:

86 IF X > 127 OR Y > 47 THEN 95
87
95 X = X − Y : Y = Y − 48
96 GO TO 3Ø

Für ein paar weitere Fingerübungen möchte ich Ihnen folgende Anregungen geben:

- Erhöhen Sie X ODER Y um mehr als 1: Der Punkt hüpft in größeren Schritten.
- Erhöhen Sie X UND Y um mehr als 1: Der Punkt verläuft zusätzlich unter einem anderen Winkel zur Horizontalen.
- Ersetzen Sie Zeile 6Ø durch CLS.
- Entfernen Sie die beiden Warteschleifen 4Ø und 8Ø: Sie haben damit die maximale Geschwindigkeit entdeckt, mit der bei diesem Gerät unter BASIC „gezeichnet" werden kann.

Lösen Sie doch einmal folgende Zeichenaufgabe: Beginnend ab der Druckposition 348 soll das Wort BASIC auf den Schirm geschrieben und durch ein Rechteck aus Leuchtpunkten eingerahmt werden!

Versuchen Sie's mit:

```
5 CLS
10 Y = 14
20 FOR X = 55 TO 66
30 SET(X,Y)
40 NEXT X
50 PRINT@348,"BASIC"
60 Y = 18
70 FOR X = 55 TO 66
80 SET(X,Y)
90 NEXT X
100 X = 55
110 FOR Y = 15 TO 17
120 SET(X,Y)
130 NEXT Y
140 X = 66
150 FOR Y = 15 TO 17
160 SET(X,Y)
170 NEXT Y
200 GO TO 200
```

Ich gebe Ihnen recht: Auch das ist ein ziemlicher Aufwand für ein bescheidenes Ergebnis!

10.6 Sie entwerfen ein Achsenkreuz

Eine weitere Übung: Sie wollen auf dem Bildschirm ein Achsenkreuz zeichnen:

```
5 CLS
10 X = 63
20 FOR Y = 0 TO 47
30 SET(X,Y)
40 NEXT Y
50 Y = 24
60 FOR X = 0 TO 127
70 SET(X,Y)
80 NEXT X
200 GOTO 200
```

Und weiter geht's: Sie zeichnen zwei Koordinaten, die Sie nach dem Muster der *Abb. 21* und *22* mit einer Skala beschriften.

Für diese und ähnliche Aufgaben ist die genaue Kenntnis der zu verwendenden PRINT- und Zeichenpositionen unerläßlich. Ich habe in den Abb. 21 und 22 von einem „VIDEO WORKSHEET" Gebrauch gemacht; von einem „Formblatt", das der Hersteller des Computers anbietet.

Zum Programm: Wir entwickeln es in einigen Teilstücken und beginnen mit der Beschriftung der Y-Achse.

```
5 CLS
20 PRINT@129, "5"
30 PRINT@257, "4"
40 PRINT@385, "3"
50 PRINT@513, "2"
60 PRINT@641, "1"
```

Wir setzen die Skalenpunkte neben die Ziffern:

```
70 X = 5
80 FOR Y = 7 TO 36 STEP 6
90 SET(X,Y)
100 NEXT Y
```

Sie zeichnen die Y-Achse:

```
110 X = 6
120 FOR Y = 1 TO 36
130 SET(X,Y)
140 NEXT Y
```

Fahren Sie mit der X-Achse fort!

```
150 Y = 36
160 FOR X = 7 TO 70
170 SET(X,Y)
180 NEXT X
```

Jetzt kommen die Skalenpunkte auf der X-Achse:

```
190 Y = 37
200 FOR X = 11 TO 65 STEP 6
210 SET(X,Y)
220 NEXT X
```

Zum Schluß schreiben wir das Programmsegment für die Beschriftung der X-Achse:

```
230 PRINT@837, "1     2     3     4     5     6     7     8     9     10"
```

Jeweils zwischen 2 Ziffern 2 Blanks setzen!

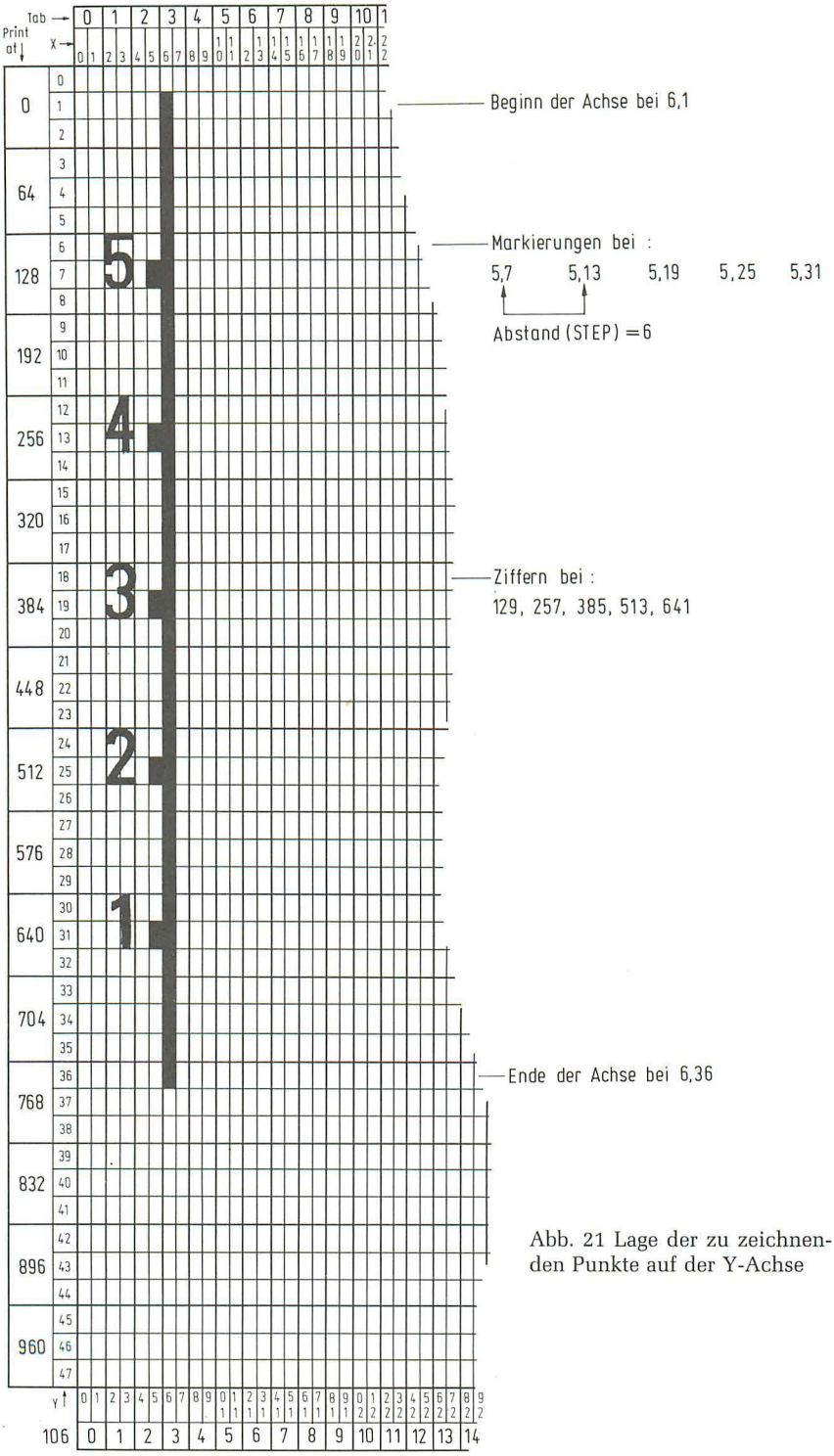


Abb. 21 Lage der zu zeichnenden Punkte auf der Y-Achse

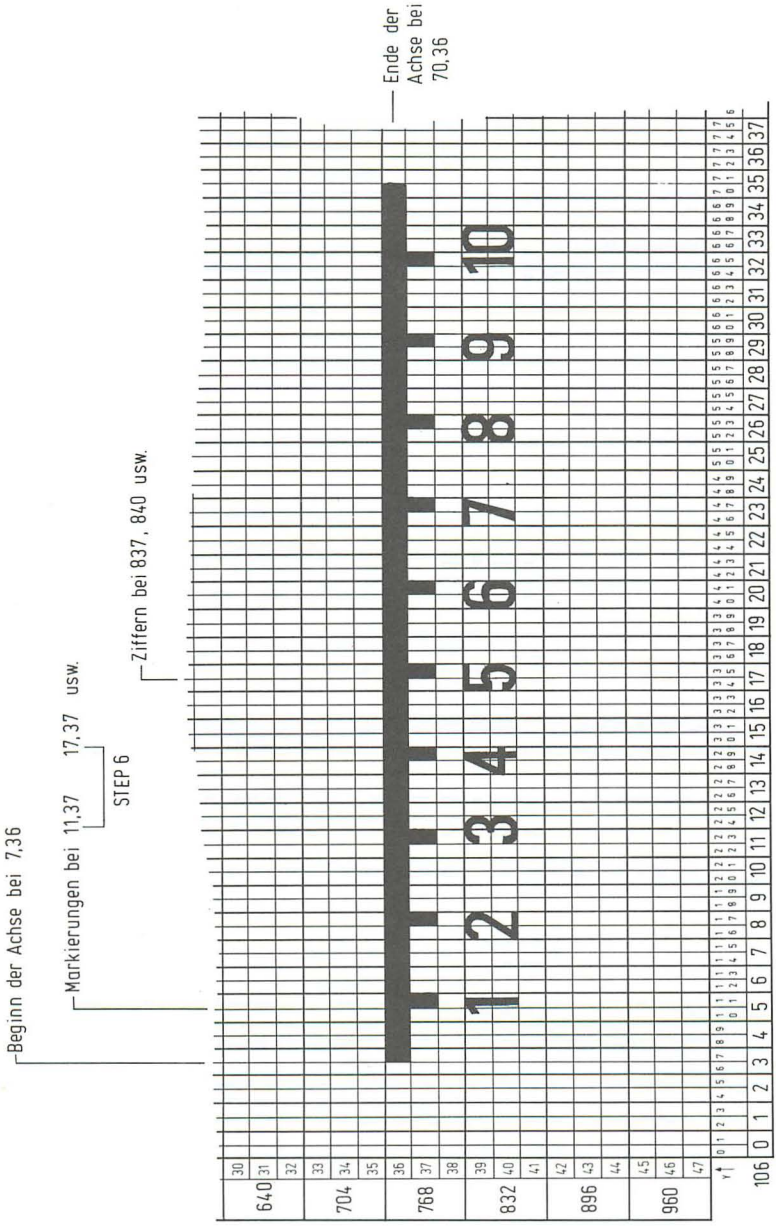


Abb. 22 Lage der zu zeichnenden Punkte auf der X-Achse

10.7 Sie entwerfen ein Balkendiagramm

In der folgenden Übung entwerfen Sie ein Balkendiagramm!

```
5 CLS
10 Y = 37
20 FOR X = 0 TO 127
30 SET(X,Y)
40 NEXT X
1000 GOTO 1000
```

Für die Beschriftung wählen wir im Abstand von jeweils 5 Blanks:

```
50 PRINT@832,"     1980     1981     1982     1983     1984     1985"
```

Zeichnen Sie nun als „Versuch" über der Jahreszahl 1982 einen „Balken" wie folgt:

```
60 FOR X = 48 TO 51
70 FOR Y = 10 TO 36
80 SET(X,Y)
90 NEXT Y,X
```

Für die nächste Aufgabe gehen Sie von folgender Annahme aus: Der Umsatz der „Computer-AG" wird für 1980 mit 1 Mio DM angenommen. Die Unternehmensleitung will eine jährliche Umsatzsteigerung für die folgenden Jahre von jeweils 5 % erzielen. Sie sollen diese Umsatzentwicklung bis 1985 in einem Balkendiagramm darstellen (*Abb. 23*).

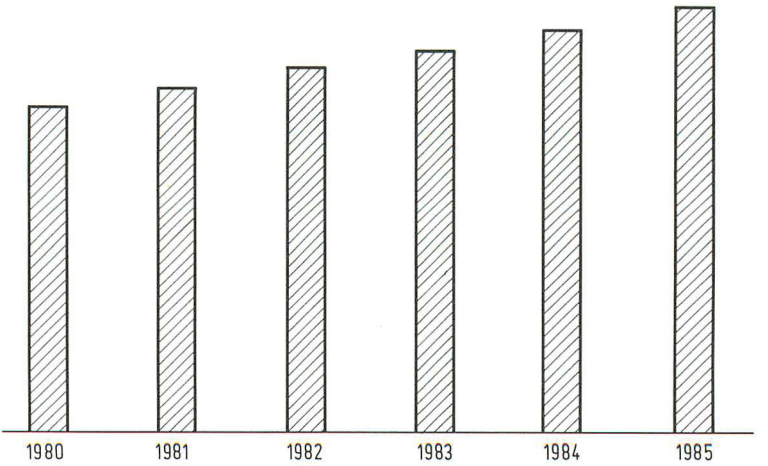


Abb. 23 So soll das Balkendiagramm aussehen

Erster Schritt:
Wir legen für 1980 = 100 % eine „Balkenlänge" von 20 Lichtpunkten zugrunde.

Da der Beginn für jeden Balken auf der Grundlinie liegt (Y = 36!!), muß dieser Balken bei 36 – 20 = 16 beginnen.

Wir überschreiben unseren letzten Versuch wie folgt:

```
60 FOR X = 12 TO 15
70 FOR Y = 16 TO 36
80 SET(X,Y)
90 NEXT Y,X
```

Zweiter Schritt:
Für 1981 soll der Umsatz 5 % höher sein. Das heißt, auch unser Balken muß 5 % länger werden.

```
110 A = 20          Das ist die Balkenlänge für 1980!
120 A = A * 1.05    Das ist die um 5 % vergrößerte Balkenlänge!
130 A = INT(A)      Das geben wir zur „Sicherheit" ein; es gibt
                    bestimmt keine Bruchteile von Bildpunkten!
140 FOR Y = (36-A) TO 36
150 B = 12
160 B = B + 18
170 C = 3
```

Nun stellen Sie das Programm – Zeilen 110...170 – so um, daß die Balken für 1981 bis 1985 in einem „Aufwasch" gezeichnet werden.

```
110 N = 0
120 A = 20
130 B = 12 : C = 3
140 A = A * 1.05 : A = INT(A)
150 B = B + 18
160 FOR Y = (36 - A) TO 36
170 FOR X = B TO B + C
180 SET(X,Y)
190 NEXT X,Y
210 N = N + 1
220 IF N > 4 THEN 1000
230 GO TO 140
```

Betrachten Sie sich Ihr Werk! Und vergleichen Sie das Programm mit einer Zeichnung, die Sie sich auf einem Arbeitsblatt angefertigt haben! Sie haben an dieser Aufgabe erkannt:

- Die Zeichenpositionen, die Anzahl der Zeichen lassen sich auch in Abhängigkeit von mathematischen Beziehungen wählen, wie das in den Zeilen 14Ø, 15Ø geschehen ist.
- Ohne Arbeitsblatt mit den „Adressen" der Bild- und PRINT-Positionen ist das Zeichnen ein mühsames Unternehmen!

Haben Sie Lust, das Bild noch etwas zu verschönern?

```
22Ø IF N > 4 THEN 24Ø
24Ø PRINT@6,"COMPUTER AG"
25Ø PRINT@7Ø,"UMSATZPROGNOSE"
```

Wenn Sie an dieser Aufgabe weiterarbeiten möchten, dann versuchen Sie doch noch folgende Varianten:
- Schreiben Sie über jeden Balken den tatsächlichen Wert in Mio. DM!
- ...oder die „Prozente"!
- Bringen Sie am rechten Bildrand eine Skala an, auf der im richtigen Maßstab steht: 1ØØ %...1Ø5 %...11Ø % usw.!
- Drehen Sie die ganze Geschichte um und lassen Sie die Balken horizontal verlaufen!

10.8 Sie zeichnen eine mathematische Funktion

Nehmen Sie an, Sie haben zwei Zahlen, A und B. Wenn beide gleich groß sind, können Sie schreiben: A = B. Wollen Sie diesen Zusammenhang in einer Grafik darstellen, dann kommt folgendes Bild (*Abb. 24*) heraus:

Versuchen Sie, dieses Bild durch den Computer zeichnen zu lassen (es wird zunächst auf die Darstellung des Achsenkreuzes verzichtet):

```
5 CLS
1Ø A = Ø
2Ø B = A
3Ø SET(A,B)
4Ø A = A + 1 : B = A
6Ø GOTO 3Ø
```

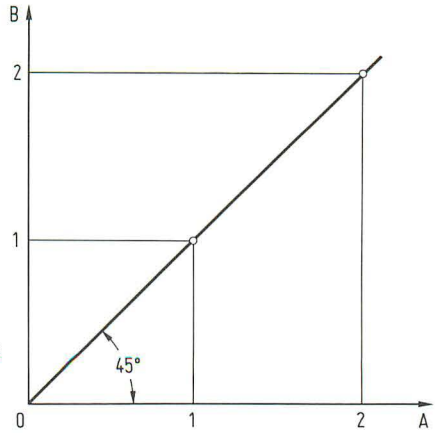


Abb. 24 Zeichnen der Beziehung A = B

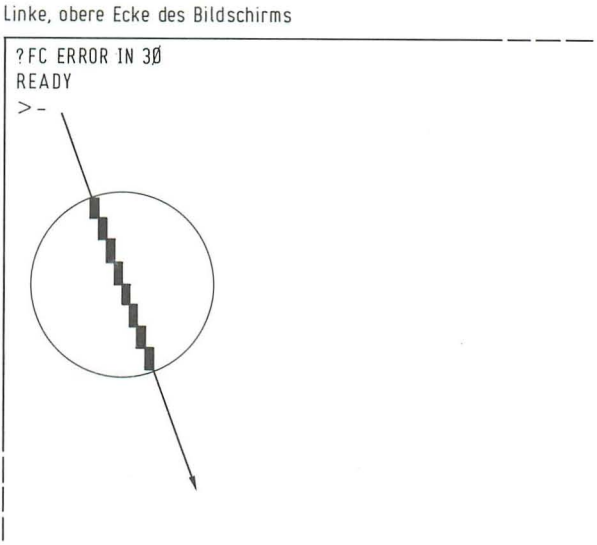


Abb. 25 ...und das hat der Computer daraus gemacht!

Das Ergebnis entspricht sicher ganz und gar nicht Ihren Erwartungen. Zählen Sie zunächst alle Punkte auf, die Ihnen überraschend und fremd vorkommen!

- Das, was der Computer gezeichnet hat, ist kein „Strich", wie Sie ihn in Abb. 24 gezeichnet hatten, sondern eine Aneinanderreihung kleiner Rechtecke, eben von Bild-„Punkten". Dies liegt ganz einfach daran, daß die Auflösung dieses Computers vergleichsweise gering ist. Wir können das nicht ändern.
- Während Sie Ihre Zeichnung – Abb. 24 – in der linken, unteren Ecke bei Ø,Ø für A,B begonnen hatten, zeichnet der Computer von links oben nach rechts unten; für ihn liegt der Bildpunkt Ø,Ø in der linken, oberen Ecke.
  Das können Sie auf gar keinen Fall akzeptieren, würde es Sie doch zwingen, Ihrem Computer zuliebe von allen Regeln der mathematischen Darstellungskunst abzuweichen (*Abb. 25*).

Hier können wir etwas tun! Schreiben Sie einmal:

```
5 CLS
1Ø A = Ø
2Ø B = 47
3Ø SET(A,B)
4Ø A = A + 1
5Ø B = B - 1
6Ø GOTO 3Ø
```

Gut so? Zwar ist in dem Programm von Ihrer Ausgangsformel A = B nichts mehr zu erkennen. Aber immerhin beginnt der Computer jetzt sein Bild links unten. Wir werden dieses Problem später lösen; fahren wir zunächst in der Aufzählung der Ungereimtheiten fort!

– In Abb. 24 hatten wir angegeben, daß die Gerade durch alle Punkte A = B unter einem Winkel von 45° zur Horizontalen verläuft. Das tut die vom Computer gezeichnete Gerade ganz offensichtlich nicht. Die Erklärung dafür finden Sie in *Abb. 26*.

  Bei a) ist ein Leuchtpunkt stark vergrößert; Sie sehen, daß es sich um ein Rechteck mit dem Seitenverhältnis von etwa 2 : 3 handelt. Da der Computer in Richtung der Diagonalen zeichnet, kommt durch dieses Seitenverhältnis die beobachtete Abweichung von 45° zustande. Zum Vergleich ist bei b) ein Lichtpunkt mit dem Seitenverhältnis 1 : 1 *angenommen*; hier verläuft die Diagonale unter einem Winkel von 45°.

– Schließlich erhalten Sie in Ihrem Computerbild oben links die Klage:
  ? FC ERROR IN 3Ø

  Das liegt daran, daß in dem Programm keine Begrenzungen für A und B eingegeben waren. Für das letzte Beispiel können Sie den Schaden beheben, wenn Sie folgende Zeilen in das Programm einfügen:

```
55 IF B = Ø THEN 7Ø
7Ø GOTO 7Ø
```

  Es sei hier angemerkt, daß die obige Fehlermeldung bei Überschreiten der Grenzen für A bzw. B nur beim TRS-80 erscheint; andere Systeme mit vergleichbaren Grafik-Fähigkeiten setzen die Variablen bei Überschreiten der Grenzen von 47 bzw. 127 automatisch wieder auf Ø,Ø.

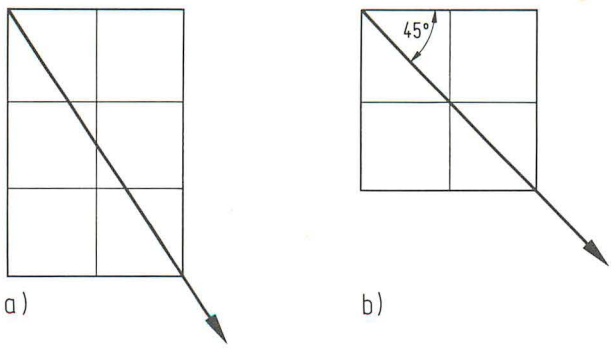


Abb. 26 Vergrößerung eines Bildpunktes

Wenden wir uns der Lösung der noch verbliebenen Probleme zu!
Entwerfen Sie zunächst das Programm für die X- und Y-Achse.

```
  5 CLS
 10 A = 5
 20 FOR B = 7 TO 31 STEP 6
 30 SET(A, B)
 40 NEXT B

 50 PRINT @ 129,"5":PRINT @ 257,"4":PRINT @ 385,"3"
 60 PRINT @ 513,"2":PRINT @ 641,"1"

 70 A = 6
 80 FOR B = 1 TO 36
 90 SET(A,B)
100 NEXT B

110 PRINT @ 837,"1 2 3 4 5 6 7 8 9 10"

120 B = 36
130 FOR A = 7 TO 70
140 SET(A,B)
150 NEXT A

160 B = 37
170 FOR A = 11 TO 65 STEP 6
180 SET(A,B)
190 NEXT A

500 GOTO 500
```

Wenn Sie einen Probelauf machen, erhalten Sie ein zufriedenstellendes Ergebnis. Bevor die Programmierarbeit fortgesetzt wird, ein paar Merkregeln:

- *Eine* Druckposition (PRINT..) enthält *sechs* Zeichenpositionen (SET...).
- Ein in eine Druckposition gedruckter Buchstabe füllt nur die *oberen vier* Zeichenpositionen aus.
- Sie dürfen beim Entwurf Ihrer Bilder Druck- und Zeichenpositionen sich nicht gegenseitig überdecken lassen, weil Sie sich dann gegenseitig auslöschen.
  Das gilt besonders für die in einer Druckposition sitzenden, beiden unteren Zeichenpositionen, die von dem gedruckten Zeichen nicht ausgefüllt werden.

60,21	61,21
60,22	61,22
60,23	61,23

Dargestellt ist die Druckposition 478

Diese umfaßt 6 Zeichenpositionen von 60,21 ... 61,23.

Ein Zeichen - z.B. Ø - würde in den Positionen 60,21; 61,21; 60,22 und 61,22 dargestellt

Abb. 27 Darstellung der Druckposition 478

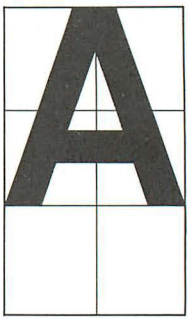


Abb. 28 In einer Druckposition nimmt ein gesetztes Zeichen nur die vier oberen „Zeichen"-Positionen ein

Die *Abb. 27* und *28* machen die Zusammenhänge deutlich.

Jetzt gilt es, in die entworfenen Koordinaten die Gerade einzuzeichnen, die der Gleichung A = B folgt. Sie erinnern sich: Ist A = Ø, dann ist auch B = Ø. In der Abb. 24 hatte die Gerade bei Ø,Ø ihren Anfang. Sie sehen ein, daß das hier auf diese Weise nicht zu bewerkstelligen ist.

Versuchen Sie, aus den Daten für die gezeichnete X-Y-Achse ein paar Anhaltspunkte zu gewinnen!

- Die Ø,Ø-Position ist in der Grafik die Position 7,35
- Der Abstand der Achsenbeschriftung ist:
  - vertikal 6 SET-Positionen
  - horizontal 6 SET-Positionen
- Der Punkt A = 1, B = 1 erscheint auf der Position 11,31.

Setzen Sie diese Angaben „mit Verstand" um, dann erhalten Sie für die zu zeichnende Gerade folgendes Programm:

```
2ØØ A = 7
21Ø B = 35
22Ø SET(A,B)
23Ø A = A + 1
24Ø B = B - 1
245 IF B = 5 THEN 5ØØ
25Ø GOTO 22Ø
```

Geschafft!

Sie haben sicher mit einiger Verwunderung erkannt, wie mühsam es ist, solche einfachen Dinge grafisch darzustellen. Besonders deshalb, weil Sie aus Erfahrung wissen, mit welcher Leichtigkeit Sie den Zusammenhang A = B in einer Tabelle „ausgedruckt" erhalten.

Der Vollständigkeit halber möchte ich erwähnen, daß die geschilderten Prozeduren nur für den TRS-80 und diesem vergleichbare Geräte gelten. Es gibt durchaus Personalcomputer – wie z. B. den APPLE – die eine wesentlich komfortablere „Computergrafik" bieten und derartige Aufgaben mit weniger Programmieraufwand erledigen.

Da das Programm soviel Mühe gemacht hat, wollen wir es noch nutzen und eine zweite Funktion hineinzeichnen, nämlich:

A = 2*B

Das heißt: ist A = 1, dann ist B = 2 usw.

Für unser „Bildprogramm" bedeutet das:

Lassen wir B um einen Bildpunkt nach oben gehen, dann muß A um zwei Bildpunkte nach rechts wandern.

Damit wir nach Zeichnen der Geraden A = B die Grenze A = 2*B zeichnen können, verlegen wir zunächst die Grenze aus der vorigen Aufgabe:

```
245 IF B = 5 THEN 3ØØ

3ØØ A = 7
31Ø B = 35
32Ø SET(A,B)
33Ø A = A + 2
34Ø B = B - 1
35Ø IF A = 65 THEN 5ØØ
36Ø GOTO 32Ø
```

10.9 Sie zeichnen eine Figur

Für die ganze „Malerei" haben Sie nur zwei spezielle Worte verwendet: SET und RESET. Daneben haben Sie in den Übungsprogrammen Kommandos, Statements, Funktionen und Programmiertechniken wie „Zählschleifen bilden" eingesetzt, die Ihnen aus anderen Aufgaben längst geläufig waren.

Wenn ich Sie im folgenden Abschnitt dahin bringe, eine Figur – und eine äußerst bescheidene dazu – zu zeichnen, dann nicht in erster Linie wegen der Zeichnerei. Es kommt mir vielmehr darauf an, Ihnen vor Augen zu führen:

- In der „Computergrafik" kommen Sie nur mit „Rechnen", mit Logik und ein wenig Knobeln zum Erfolg.

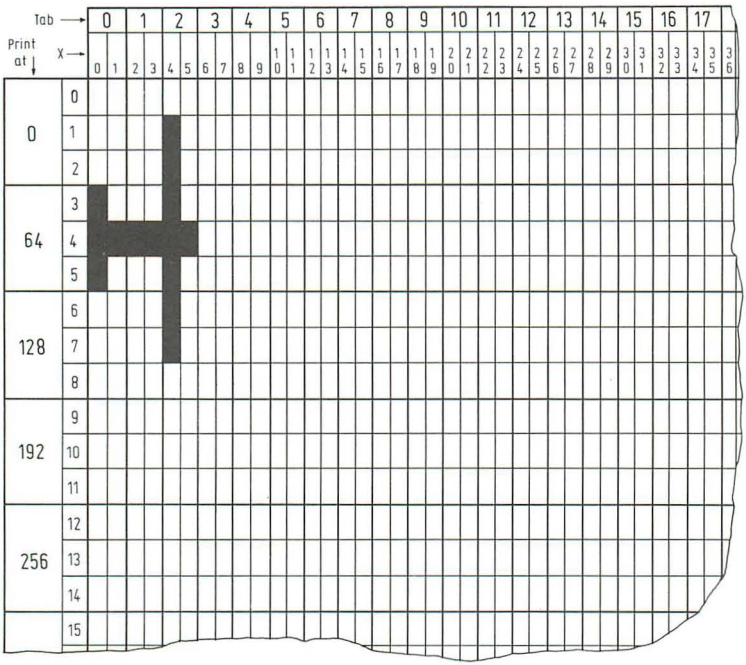


Abb. 29 Diese Figur soll gezeichnet werden

– Sie verwenden dabei BASIC-Worte und Programmiertechniken, die Sie in ganz anderen Aufgaben schon hinreichend eingeübt haben.

Und nun zu der Figur! *Abb. 29* zeigt, *was* zu zeichnen wir uns vornehmen. Das soll ein „Flugzeug" darstellen; Sie können's aber auch für eine Taube halten.

Beachten Sie das der Figur unterlegte Bildschirm-Raster! Aus Gründen, die Sie im Verlauf der Aufgabe durchschauen werden, soll die Programmierung mit der READ-DATA-Technik vorgenommen werden.

```
 5 CLS
1Ø READ X,Y
2Ø IF X = 999 THEN 1ØØØ
3Ø SET(X,Y)
4Ø GOTO 1Ø

1ØØØ GOTO 1ØØØ
```

Als „Daten" geben Sie nun die in Abb. 29 ausgefüllten Bildpunkte ein; und zwar für jeden Punkt zunächst den X-, danach den Y-Wert. Die Zeile 2Ø ist eingefügt, um den sonst unvermeidlichen Ausdruck ..."OD ERROR IN.." zu vermeiden. Also:

```
5ØØ DATA Ø,3,Ø,4,Ø,5,1,4,2,4,3,4,4,1,4,2,4,3,4,4,4,5,4,6,4,7,5,4,999,999
```

Gefällt Ihnen das Flugzeug? Dann wollen wir es „fliegen" lassen!

10.10 Die Figur bewegt sich!

Wissen Sie, wie man einen Zeichentrickfilm macht?

Man zeichnet mit einer Filmkamera ein Bild von der positionierten Figur auf... Bewegt die Figur ein wenig vorwärts... Nimmt dieses Bild auf... usw.
Bei der Vorführung des auf diese Weise entstandenen Filmes ergibt sich der Eindruck einer bewegten Figur. Ab ca. 16 Einzelbildern pro Sekunde kann das menschliche Auge die Einzelbilder nicht mehr als solche erfassen; es entsteht der Eindruck einer fließenden Bewegung. Unter 16 Bildern/Sekunde – sagen wir bei 5/Sek. – „hüpft" der fotografierte Gegenstand von einer Position zur nächsten. Und ganz genau so können Sie auf Ihrem Bildschirm den Eindruck eines bewegten Bildes erzeugen.

Das bedeutet:

Das in Abb. 29 gezeichnete Flugzeug wird wieder gelöscht. Dann – um eine Position in X-Richtung verschoben – neu gezeichnet. Wieder gelöscht. Und so weiter. Versuchen Sie's mal!

```
20 IF X = 999 THEN 50
50 RESTORE
60 READ X,Y
80 RESET(X,Y)
90 GOTO 60
```

Sie sehen: kaum ist das Flugzeug gezeichnet, wird es wieder gelöscht.

Ich hoffe, daß Ihnen inzwischen auch klar geworden ist, warum hier mit READ-DATA gearbeitet wird? Weil wir die einmal gespeicherten Daten für die Bildpunkte immer wieder verwenden können.

Wenn Sie die Zeitspanne zwischen „Zeichnen" und „Ausradieren" vergrößern wollen, können Sie eine Warteschleife einfügen, etwa:

```
50 FOR Z = 1 TO 100 : NEXT
55 RESTORE
```

Probieren Sie das mal mit verschiedenen langen Schleifen! Und dann löschen Sie 50 wieder!

Die nächste Aufgabe in unserem „Zeichentrickfilm" besteht darin, ein neues Flugzeug zu zeichnen. Allerdings mit einer um jeweils 1 erhöhten X-Position, da wir es horizontal über den Bildschirm bringen wollen.

Das scheint einfach zu sein:

```
...READ X,Y
...X = X + 1
...SET(X,Y)
```

Bevor Sie das so versuchen, sollten Sie folgendes bedenken: Die X,Y-Positionen stehen in der DATA-Liste und lassen sich *dort* nicht verändern. Das zweite Bild ließe sich zwar nach dem Ansatz oben noch zeichnen. Aber was wäre mit dem dritten? Da würde ja X = X + 1 + 1 oder X = X + 2 sein.

Wir müssen in das Programm einen „Bilderzähler" einführen, der nach jedem gezeichneten – oder gelöschten – Bild die Zahl der Bilder um 1 erhöht.

Darf ich Ihnen einmal aufzeichnen, wie ich das meine (*Abb. 30*)?

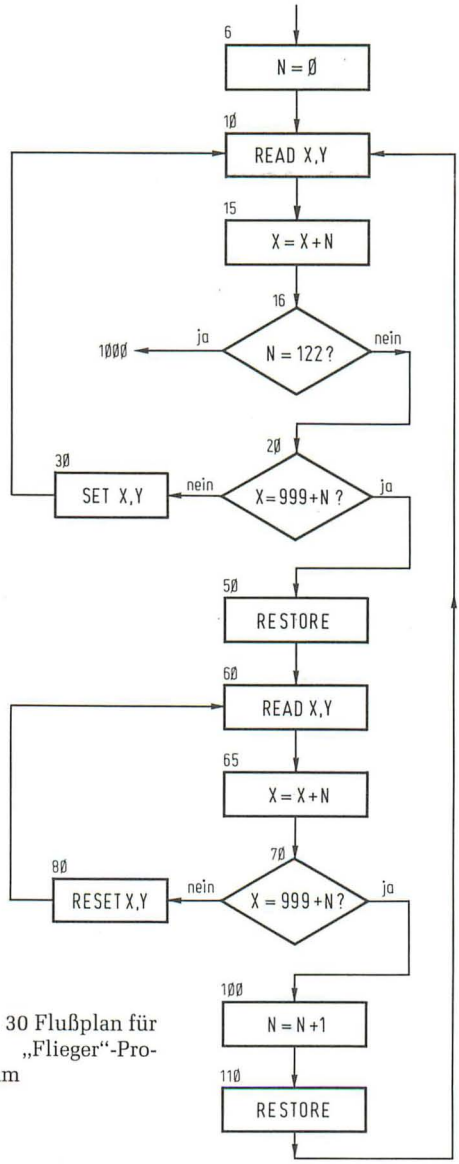


Abb. 30 Flußplan für das „Flieger"-Programm

Das führt zu folgendem Programm:

```
  5 CLS
  6 N = Ø
 1Ø READ X,Y
 15 X = X + N
 16 IF N = 122 THEN 1ØØØ
 2Ø IF X = 999 + N THEN 5Ø
 3Ø SET(X,Y)
 4Ø GOTO 1Ø

 5Ø RESTORE
 6Ø READ X,Y

 65 X = X + N
 7Ø IF X = 999+N THEN 1ØØ
 8Ø RESET(X,Y)
 9Ø GOTO 6Ø
1ØØ N = N + 1
12Ø RESTORE
13Ø GOTO 1Ø
```

Folgende Erklärungen bin ich Ihnen noch schuldig:

- 6 N = Ø; N ist der erwähnte „Bilderzähler"
- 15,65 X = X + N; Der Bildpunkt wandert um X + N nach rechts
- 16 IF N = 122... Schon im ersten Bild belegte die „Nase" des Flugzeuges die Position X = 5. Soll das Bild bei X = 127 verschwinden, dann darf N nur noch 127 − 5 = 122 erreichen
- 2Ø,7Ø IF X=...
  Das „Datum" für „Ende des Bildes" ist X = 999. Da auch 999 jeweils um 1 erhöht wird, müssen wir die Grenze entsprechend 999 + N „vor uns herschieben"
  N = N + 1; der Bildzähler wird um 1 erhöht.

Das Ergebnis Ihres Fleißes ist nicht berauschend. Insbesondere von „Fliegen" kann wohl keine Rede sein. Immerhin dauert es fast zwei Minuten, bis das Bild den rechten Rand erreicht hat.

Sie sind hier zum ersten Male überdeutlich an ein Handikap von BASIC gestoßen, genauer: an die Geschwindigkeit, mit der BASIC-Programme abgearbei-

tet werden. In unserem Beispiel dauert es einfach solange, bis 122 Bilder „auf-" und „abgebaut" sind.

Ich sollte der Vollständigkeit halber erwähnen, daß auch der TRS-80 Mittel und Wege kennt, das Flugzeug wirklich fliegen zu lassen. Einen interessanten Weg werden Sie noch kennenlernen; ich bitte noch um ein Kapitel Geduld!

Ändern Sie einmal Ihre DATA-Zeile zu:

```
500 DATA 0,4,1,4,2,4,999,999
```

Und betrachten Sie das Bild meinetwegen als fliegende Unterasse. Experimentieren Sie trotz Ihrer Enttäuschung ein wenig!

- wählen Sie in 100: N = N + 2, oder ...3
- verwenden Sie statt RESET einmal CLS

10.11 Sie durchsuchen den Bildschirm

Gegen Ende unserer Zeichenstunde will ich Ihnen noch eine spezielle Funktion vorstellen, mit deren Hilfe Sie herausfinden können, ob ein angesprochener Lichtpunkt „an" oder „aus" ist.

Zuvor sollen Sie jedoch eine Preisfrage beantworten!

- Der Bildschirm hat 16 Zeilen zu je 64 Zeichen; das ergibt 1024 PRINT-Positionen...
- Jede PRINT-Position hat 6 BILDPUNKTE, so daß Ihnen zur Verfügung stehen:
  128 Bildpunkte auf der Horizontalen
  48 Bildpunkte auf der Vertikalen
  6144 Bildpunkte insgesamt
- Die Preisfrage:
  Wie kommt man von einem mit X,Y definierten Bildpunkt auf die PRINT-Position, in der sich dieser Bildpunkt befindet?

Nun, knobeln Sie! Wenn Sie das nicht herausbekommen, verrate ich Ihnen das Ergebnis meines Nachdenkens:

PRINT-Position = (INT(Y/3)*64) + INT(X/2)

Damit Sie das auch glauben, prüfen Sie das bitte mit folgendem Programm:

```
 5 CLS
1Ø INPUT"BITTE X DES BILDPUNKTES EINGEBEN !";X
2Ø INPUT"BITTE Y DES BILDPUNKTES EINGEBEN !";Y
3Ø U = INT(Y/3)*64
4Ø W = INT(X/2)
5Ø Z = U + W
6Ø PRINT "DER BILDPUNKT X,Y =";X;",";Y;"LIEGT IN DER ";Z;" TEN PRINT-
          POSITION."
```

Nach diesen Vorübungen zur „Lichtpunktsuche"! Setzen Sie einen Bildpunkt:

```
 5 CLS
1Ø X=62 : Y = 22
2Ø SET(X,Y)
```

Die Suchfunktion heißt (beim TRS-80): POINT...

- Ist der angesprochene Bildpunkt „hell" = eingeschaltet, dann antwortet POINT... mit −1
- Ist der angesprochene Bildpunkt „dunkel", dann antwortet POINT... mit Ø

Probieren Sie's aus!

```
4Ø PRINT POINT(X,Y)
```

Und zum Beweis, prüfen Sie den Punkt daneben!

```
4Ø PRINT POINT(X+1,Y)
```

Stimmt's? Eine weitere Übung:

```
4Ø IF POINT(X,Y)=−1 THEN PRINT"DER PUNKT";X;",";Y;"IST HELL"
5Ø IF POINT(X+1,Y)=Ø THEN PRINT"DER PUNKT";X;",";Y;"IST DUNKEL"
6Ø IF POINT(X+2,Y)=−1 THEN PRINT "HELL" ELSE PRINT "DUNKEL"
```

Oder:

```
6Ø IF POINT(X+2,Y)= −1 THEN PRINT "HELL" : END
7Ø PRINT "DUNKEL"
```

Ändern Sie das Programm doch noch einmal wie folgt ab:

```
 40 FOR X = 0 TO 127
 42 FOR Y = 0 TO 47
 50 IF POINT(X,Y) = -1 THEN 100
 60 NEXT Y,X
100 PRINT „EINGESCHALTET!"
```

Sie erkennen, daß wir mit diesem Programm nun den Bildschirm – beginnend bei der Position 0,0 zeilenweise durchforschen, ob wir den gesetzten Bildpunkt finden. Haben wir ihn erwischt, dann wollen wir mit dem folgenden Zusatzprogramm an die Position X,Y – im Beispiel 62,22 – das Wort „GETROFFEN!" erscheinen lassen. Zur Lösung machen Sie sich das Ergebnis der Übung auf Seite 107 zunutze.

```
100 U = INT(Y/3)*64
110 W = INT(X/2)
120 Z = U + W
150 PRINT @ Z,"GETROFFEN!"

160 PRINT @ 768,"DER GETROFFENE LICHTPUNKT IST :";X;",";Y
170 PRINT @ 832,"ER LIEGT IN DER PRINT-POSITION";Z
```

Achtung, verzagen Sie nicht! Der Computer benötigt mehr als 30 Sekunden, um den gesetzten Punkt zu finden!

10.12 Sie arbeiten mit dem Zeichensatz Ihres Computers

Bitte lesen Sie noch einmal auf Seite 46 nach, was dort zu der Funktion CHR$(...) steht. Danach sollten Sie mit dem Programm auf Seite 46 den Zeichensatz Ihres Computers durchmustern.

- Der TRS 80 bietet für die ASCII-Codenummern 128...191 einen Satz von 64 grafischen Zeichen; diese habe ich Ihnen in der *Tabelle 6* aufgelistet. Behalten Sie im Gedächtnis, daß jedes Zeichen eine der 1024 *Buchstaben*-Positionen auf dem Bildschirm ausfüllen kann. Die weiteren Ausführungen dieses Kapitels gelten für den TRS 80.

Tabelle 6: Grafik-Zeichensatz des TRS 80

ASCII-Code	Zeichen	ASCII-Code	Zeichen	ASCII-Code	Zeichen	ASCII-Code	Zeichen
128		144		160		176	
129		145		161		177	
130		146		162		178	
131		147		163		179	
132		148		164		180	
133		149		165		181	
134		150		166		182	
135		151		167		183	
136		152		168		184	
137		153		169		185	
138		154		170		186	
139		155		171		187	
140		156		172		188	
141		157		173		189	
142		158		174		190	
143		159		175		191	

Abb. 31 Ausgaben für CHR. $ (...)

- Besitzen Sie ein anderes Gerät als den TRS 80, dann entdecken Sie unter Umständen andere Zeichen unter anderen Codenummern. Wenden Sie dann die Ausführung in diesem Kapitel sinngemäß an.
- Hat Ihr Computer keine Grafikzeichen zu bieten, dann begnügen Sie sich mit der Durchsicht dieses Kapitels.

Geben Sie einmal ein:

```
? CHR$(65)
? CHR$(61)
? CHR$(191)
```

Die *Abb. 31* zeigt Ihnen, was jeweils auf dem Bildschirm erscheint; mit CHR$(191) haben Sie also eines der 64 Grafikzeichen aufgerufen.

Wie Sie sehen, zaubern Sie die Grafikzeichen ebenso mit PRINT... auf den Schirm wie ein gewöhnliches, alphanumerisches Zeichen. Es gilt dabei die Eigenschaften von PRINT... zu beachten. Damit Sie damit vertraut werden, schlage ich zunächst ein paar Fingerübungen vor.

```
  5 CLS
 1Ø PRINT CHR$(191) : PRINT
 2Ø PRINT CHR$(191),CHR$(191),CHR$(191),CHR$(191) : PRINT
 3Ø PRINT CHR$(191);CHR$(191);CHR$(191);CHR$(191) : PRINT
 4Ø PRINT @ 544,CHR$(191) : PRINT @ 544+64,CHR$(191) : PRINT
 5Ø PRINT TAB(Ø)CHR$(191);TAB(63) CHR$(191)

1ØØ GO TO 1ØØ
```

Sie können sich nun daran wagen, anhand der *Abb. 32* ein „Bild" zusammenzusetzen. Wie üblich, ist dabei ein Formblatt mit den einzelnen Bildpositionen nützlich. Das „Bild" bringen Sie wie folgt auf den Schirm:

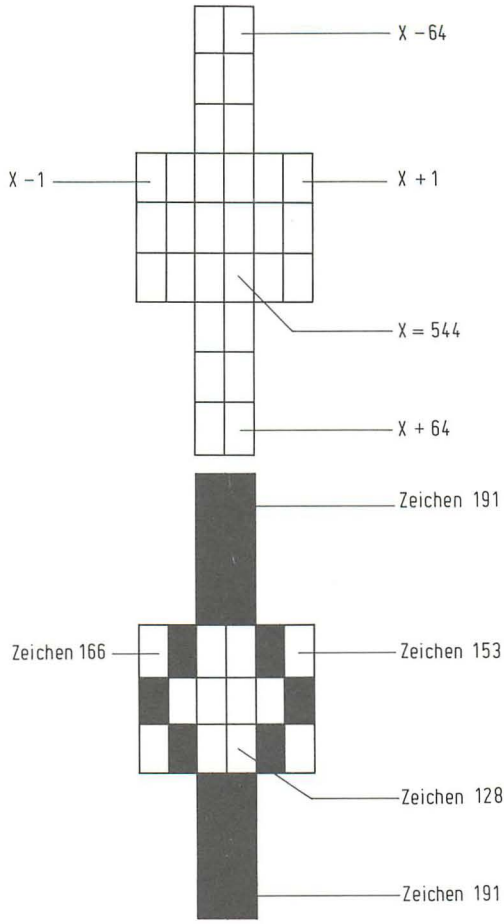


Abb. 32 Zeichnen einer Figur mit Grafik-Zeichen

```
 10 CLS
 20 X = 544
 30 PRINT @ X-64, CHR$(191)
 40 PRINT @ X-1,CHR$(166);CHR$(128);CHR$(153)
 50 PRINT @ X+64,CHR$(191)
100 GO TO 100
```

Spätestens jetzt werden Sie erkennen, daß der Aufbau des „Bildes" wesentlich schneller erfolgt als mit SET(...). Und das sollte Ihnen Mut machen, Ihre verunglückte „Flugvorführung" von vorhin mit besserem Ergebnis zu wiederholen. Ergänzen Sie zunächst das obige Programm wie folgt:

```
60 FOR X = 1 TO 100 : NEXT
70 CLS
80 FOR X = 1 TO 100 : NEXT
90 GO TO 20
```

Mit den folgenden Mini-Programmen möchte ich nichts weiter als Ihre Kreativität anregen – deshalb störe ich Sie dabei auch nur äußerst sparsam mit Kommentaren!

```
  5 CLS
 1Ø A$=CHR$(161)+CHR$(162)+CHR$(163)
 2Ø BS=CHR$(166)+CHR$(166)+CHR$(166)+CHR$(166)
 3Ø PRINT A$ : PRINT
 35 PRINT A$+B$ : PRINT
 5Ø PRINT A$+A$+B$+B$ : PRINT
 6Ø FOR Z = 1 TO 5
 7Ø PRINT A$+B$
 8Ø NEXT Z
1ØØ GOTO 1ØØ
```

```
 5 CLS
1Ø A = 128
15 D = 544
2Ø B = RND(64)
3Ø C = A + B
4Ø PRINT @ D,CHR$(C)
5Ø D = D + 1
6Ø FOR X = 1 TO 1ØØ : NEXT
7Ø GOTO 2Ø
```

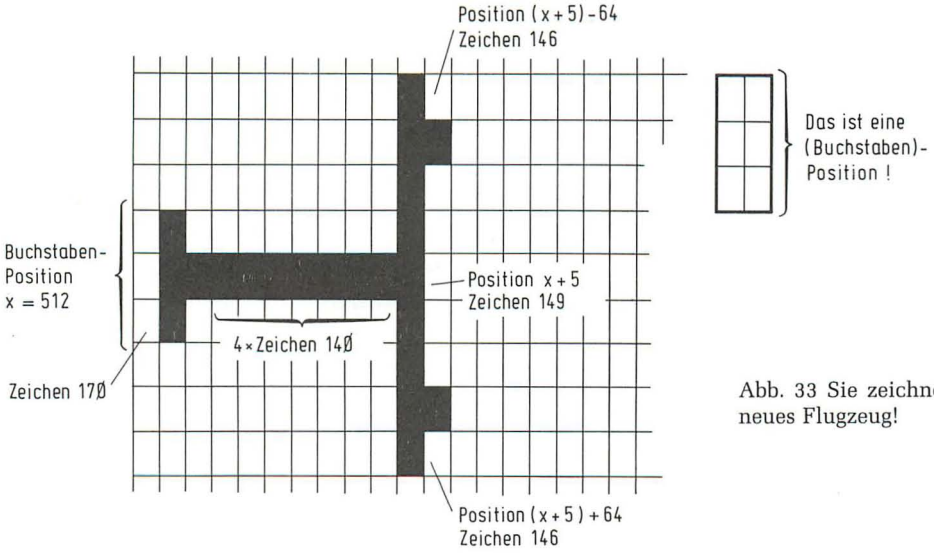


Abb. 33 Sie zeichnen ein neues Flugzeug!

```
 5 CLS
1Ø A = 128 : B = RND(64) : C = A + B
2Ø X = RND(1Ø24)
3Ø PRINT @ X,CHR$(C)
4Ø FOR X = 1 TO 1ØØ NEXT
5Ø GOTO 5
```

Nun zu Ihrem „Flieger"! Betrachten Sie die *Abb. 33*, dieses Fluggerät gilt es, auf den Schirm zu zeichen und „fliegen zu lassen"! In Zeile 4Ø finden Sie u. a. STRING$(...); schauen Sie die Bedeutung noch einmal auf Seite 52 nach!

```
 1Ø CLS
 2Ø X = 512
 3Ø PRINT @ (X+5)−64,CHR$(157)
 4Ø PRINT @ X,CHR$(17Ø);STRING$(4,14Ø);CHR$(149)
 5Ø PRINT @ (X+5)+64,CHR$(157)
 6Ø X = X + 1
 7Ø IF X + 5 = 575 THEN 1ØØ
 8Ø CLS
 9Ø GO TO 3Ø
1ØØ CLS : END
```

Noch etwas schneller geht's mit:

```
5 A$ = CHR$(17Ø)+STRING$(4,14Ø)+CHR$(149)
4Ø PRINT @ X,A$
```

Und noch einige Spielereien:

```
 5 CLEAR 1ØØØ
1Ø CLS
2Ø FOR X = 1 TO 96Ø STEP 64
3Ø PRINT @ X,STRING$(64,153)
4Ø NEXT
5Ø GO TO 5Ø
```

Löschen Sie die Zeile 5 und studieren Sie die Wirkung. Sie „handeln" hier mit Strings!

```
 1Ø CLS
 2Ø Y = 25
 3Ø FOR X = Ø TO 127
 4Ø SET(X,Y)
 5Ø NEXT
 6Ø X = 64
 7Ø FOR Y = Ø TO 47
 8Ø SET(X,Y)
 9Ø NEXT
1ØØ GO TO 1ØØ
```

```
 1Ø CLEAR 1ØØØ
 15 CLS
 2Ø FOR X = 31 TO 31 + 15*64 STEP 64
 3Ø PRINT @ X,CHR$(149)
 4Ø NEXT
1ØØ GO TO 1ØØ
```

```
 1Ø CLEAR 1ØØØ : CLS
 2Ø PRINT @ 512,STRING$(64,176)
1ØØ GO TO 1ØØ
```

Knobeln Sie aus, warum das folgende nicht gelingen will!

```
 1Ø CLEAR 1ØØØ : CLS
 2Ø FOR X = 31 TO 31 + 15 * 64 STEP 64
 3Ø PRINT @ X,CHR$(179) : NEXT
 5Ø PRINT @ 512,STRING$(64,176)
1ØØ GO TO 1ØØ
```

Und warum geht das?

```
 1Ø CLEAR 1ØØØ : CLS
 2Ø PRINT @ 512,STRING$(64,176)
 3Ø X = 62
 4Ø FOR Y = Ø TO 47
 5Ø SET(X,Y) : NEXT
1ØØ GO TO 1ØØ
```

Bevor der Ernst von BASIC wieder beginnt, entspannen Sie sich ein wenig mit dem folgenden Programm. Sie – oder Ihr Computer – werden damit zum „freien

Künstler"! In den Zeilen 1Ø...5Ø habe ich – jeweils durch zwei Sternchen getrennt – mehrere Variable zur Auswahl hingeschrieben; Sie wählen natürlich für jeden Lauf nur eine davon. Viel Spaß!

```
   5 CLS
  1Ø A = 65 ** 6Ø ** 5Ø ** 5Ø
  2Ø B = 25 ** 2Ø ** 2Ø ** 25
  3Ø C =  7 **  6 **  5 ** RND(7)
  35 X = .Ø3 ** .Ø25 ** .Ø5 ** .Ø3
  4Ø D = 3 ** 2 ** 4 ** RND(3)
  45 P = 3.1416
  5Ø PRINT"GEBEN SIE NACH AUFFORDERUNG"
  6Ø PRINT"ZWEI GANZE ZAHLEN ZWISCHEN 2 UND 6 EIN!"
  7Ø PRINT"BEISPIEL : 3 , 5"
  8Ø INPUT E,F
 1ØØ CLS
 15Ø FOR S = Ø TO 2*P STEP X
 16Ø P = F * SIN(E * S)
 17Ø X = P * C * COS(S)+A
 18Ø Y = P * D * SIN(S)+B
 2ØØ SET(X,Y) : NEXT S
1ØØØ GO TO 1ØØØ
```

Und damit endgültig: Schluß mit der Zeichenstunde!

11 Die Grenzen von BASIC

In der vergangenen „Malstunde" hatten Sie ausgiebig Gelegenheit, dem Computer bei der Arbeit zuzusehen. Wenn Sie – etwa im nächsten Abschnitt – an einem Sortierproblem arbeiten lassen, vergeht zwischen RUN und der Computer-Ausgabe eine Zeitspanne, die Sie deutlich bis ungeduldig registrieren. An dieser vermeintlichen Langatmigkeit ist Ihr Computer unschuldig. Schuld an der langen Ausführungszeit ist die von Ihnen sicher inzwischen geschätzte Programmiersprache BASIC.

BASIC ist eine interpretierende – eine „übersetzende" – Sprache, die streng nach aufsteigenden Zeilennummern vorgeht und Zeile für Zeile untersucht, was zu tun ist. Ist etwas zu drucken ...zu berechnen ...zwischenzuspeichern?
Diese Vorgehensweise macht BASIC unter den problemorientierten Programmiersprachen zu einer relativ langsamen Sprache und – zumindest beim ersten Hinsehen – für manche Anwendungsfälle ungeeignet.

Dafür ein Beispiel:
Stellen Sie sich vor, Sie wollen in einer Maschinensteuerung die Drehzahl einer Maschine messen, die mit 3600 U/min läuft. Sie haben einen Drehzahlgeber an der Maschine, der pro Umdrehung 10 Impulse liefert. Ihr Computer muß nun in der Lage sein:

...die einzelnen Impulse zu zählen...
...sie in Beziehung zur Maßeinheit „Minute" zu bringen...
...Zwischenwerte zu speichern...
...die aktuelle Drehzahl anzuzeigen...
...in Abhängigkeit von einer vorgegebenen Drehzahl irgend etwas auszulösen...
usw. usw.

Ihr Computer würde eine Aufgabe dieser Art lässig bewältigen – allerdings nicht mit einem in BASIC geschriebenen Programm.

Sollten Sie vor derartigen Problemen stehen, dann kann ich Ihnen nur einen unzulänglichen Rat geben:

– Beschäftigen Sie sich mit der „Maschinen-Sprache"... oder, falls Ihnen das zu mühsam ist,
– beschäftigen Sie sich mit „Assembler"...

Dies ist ein Buch über BASIC – und dabei wollen wir bleiben. Für Probleme der oben geschilderten Art hat der Hersteller Ihres Computers Unterstützung in Form von Literatur, speziellen ROMs oder sonstiger Software für Sie bereit. Sie werden sich dann mit einer neuen Materie beschäftigen müssen. Ihren Computer können Sie jedoch zum Glück behalten!

Was Sie außer Mühe erwartet, sei kurz in zwei Zahlen dargestellt:
- Ein in Assembler geschriebenes Programm läuft bis zu 300mal schneller als ein in BASIC geschriebenes...
- Ein Programm, das in BASIC 24 K „lang" ist, benötigt in Assembler knapp 5 K!

Zum Schluß ein paar Worte zum Trost: Sie sollten aus dem oben Gesagten nicht schließen, BASIC sei für Programme zur Meß- und Steuerungstechnik ungeeignet. Wenn Sie daran denken, daß Sie die Heizanlage Ihres Hauses überwachen wollen, dann geht das durchaus mit einem BASIC-Programm. Weil Sie bei einer derartigen Aufgabe mit wenigen Messungen pro Minute auskommen – und das schafft auch BASIC.

[1]) Siehe Anmerkung Seite 124

12 Wie spart man Speicherplatz?

Bei den von uns verwendeten Übungsprogrammen von geringem bis mittlerem Umfang sind Sie vermutlich noch nicht an die Grenzen der Speicherkapazität Ihres Computers gestoßen. Das kann sich ändern, wenn später Ihre eigenen Programme länger und komplexer werden; vor allem, wenn Sie mehr Daten verarbeiten, als wir das in den Übungsprogrammen getan hatten.

Bevor wir uns mit dem Sparen beschäftigen, wollen wir uns die Frage stellen: „Was ist das überhaupt: Speicherplatz?" Gemeint ist damit die Kapazität des Arbeitsspeichers, den Ihnen Ihr Computer zur freien Verfügung hält; und zwar für Programme und Daten. Home-Computer werden meist mit 4-K- oder 16-K-Speicher geliefert; in der Regel lassen sich durch zusätzliche ICs oder Speicherplatinen insgesamt 64 K Speicherplatz schaffen. Dabei steht 1 K für 1 Kilobyte = 1024 Byte.

Dabei gilt „freie Verfügung" nur mit Einschränkung. Von den insgesamt 64 K Speicherplatz, die ein Mikrocomputer ohne besondere Tricks ansprechen – „adressieren" – kann, benötigt er bis zu 20 K für sein eigenes Betriebssystem, für den BASIC-Interpreter usw. Es kann deshalb schon von Nutzen sein, bei der Programmierung auf möglichst „speichersparende" Programmierung zu achten.

Damit Sie ein Gefühl dafür bekommen, wieviel Speicherplatz Programme in Anspruch nehmen können, möchte ich Ihnen ein paar Angaben dazu machen.

- Eine numerische Variable benötigt sechs Byte – zwei für den Namen, vier Byte für den Wert.
- String-Variable benötigen zusätzlich ein Byte für jedes Zeichen des String.
- Indizierte Variable benötigen mindestens zwölf Byte; zwei für den Namen, zwei für die Größe der Matrix, zwei für die Zahl der Dimensionen und vier Byte für jedes Element.
- Jede FOR...NEXT...-Schleife benötigt bis zu 22 Byte.
- Jede Programmzeile schluckt fünf Byte – unabhängig davon, wie lang sie ist. Zwei Byte davon werden für die Zeilennummer gebraucht.

Lassen wir's dabei bewenden. Immerhin – wenn Sie wissen, was etwas kostet, können Sie auch abschätzen, was Sie einsparen können...

Es folgen ein paar Regeln, bei deren konsequenter Befolgung Sie Speicherplatz sparen oder den vorhandenen Speicher kostenlos „erweitern" können.

1. Schreiben Sie soviele Anweisungen wie möglich in eine Programmzeile. Bedenken Sie, daß dieses Buch in dieser Frage *kein* Vorbild sein will. Ich habe

aus Gründen der Programm-Übersichtlichkeit gegen diese und einige der folgenden Regeln bewußt gesündigt!
Also: statt

```
 5 A = Ø
1Ø B = Ø
15 C = Ø
2Ø D = Ø
```

schreiben Sie:

```
5 A = Ø : B = Ø : C = Ø : D = Ø
```

Während ich oben 32 Byte benötige, brauchen Sie nach meinem Vorschlag nur 2Ø Byte. Sie sparen also 12 Byte = 37,5 %!

2. Vermeiden Sie unnötige Zwischenräume in Ihrem Programm! Auch in diesem Punkt war ich Ihnen kein gutes Beispiel!
3Ø PRINT A + B, C + D
Diese „Schreibe" kostet 15 Byte; Sie sparen 2 Byte = 13 %, wenn Sie's so machen:
3ØPRINTA+B,C+D

3. Wenn Ihr Programm dokumentiert ist und Sie darauf verzichten können, löschen Sie alle REM aus Ihrem Programm.
Die Zeile:
5Ø REM A UND B SIND ZAEHLER
benötigt satte 27 Byte!

4. Es ist nicht notwendig, Ihr Programm mit END abzuschließen!
1ØØ END
benötigt 6 Byte!

5. Machen Sie – wann immer möglich – Gebrauch von Unterprogrammen!

6. Wenn Sie durch Probelauf wissen, daß Ihre FOR...NEXT...-Schleifen korrekt abgeschlossen sind, entfernen Sie bei NEXT die Index-Variable!
Statt: 1ØØ FOR X = 1 TO 1ØØØ : NEXT X einfach:
1ØØ FOR X = 1 TO 1ØØØ : NEXT

7. Verwenden Sie Variable statt Konstanten!
Bei Kreisberechnungen benötigen Sie beispielsweise die Größe π = 3.1416...
Rechnen Sie jetzt in Ihrem Programm:
..."FLAECHE =";2*R*3.1416, dann verwenden Sie 3.1416 als Konstante.
Schreiben Sie aber:
1Ø P = 3.1416
2Ø ..."FLAECHE =";2*R*P, dann verwenden Sie π als Variable P, der Sie den Wert 3.1416 zugewiesen haben. Je öfter Sie nun in einem Programm der obigen Art mit Hilfe von P Rechnungen durchführen lassen, um so größer ist die Speicherplatz-Ersparnis.

8. Verwenden Sie – wenn immer möglich – FOR...NEXT...-Schleifen anstelle von Zählschleifen!
9. Wenn Ihr Programm das erlaubt, rechnen Sie mit INT-Variablen!
   Beim TRS-80 können Sie das einfach machen mit: DEFINT A – Z; bei anderen Geräten müssen Sie allen Variablen, die Sie als Ganzzahlvariable behandeln lassen wollen, das Zeichen % anhängen.
   Also A%=... oder Z9%=...

Die Frage des Platzbedarfes im Speicher für ein Programm hängt eng zusammen mit der Frage, wie schnell es vom Computer abgearbeitet wird. Dies möchte ich Ihnen an zwei Beispielen demonstrieren.

Beispiel 1:
Es folgt ein einfaches Sortierprogramm, das Zahlen in aufsteigender Reihenfolge sortiert und ausgibt. Das Programm ist zunächst in der für dieses Buch typischen, übersichtlichen Darstellungsweise geschrieben. Es benötigt 699 Byte. Gibt man im Eingabeteil 50 Zahlen ein, die wahllos zwischen Ø und 999 liegen, so benötigt der Computer 25 Sekunden für das Sortieren von Ø...999.

Und hier ist das Programm:

```
5 DIM B(1ØØ) : CLS
1Ø INPUT"WIEVIEL ELEMENTE ?";N
2Ø FOR I = 1 TO N
3Ø INPUT" B =";B(I)
4Ø NEXT I
```

Nach diesem „Eingabeteil" folgt das eigentliche Sortierprogramm:

```
1ØØ Z = N - 1
11Ø FOR I = 1 TO N - 1
12Ø FOR Y = 1 TO Z
13Ø IF B(I) < B(I+Y) THEN 17Ø
14Ø H = B(I)
15Ø B(I)=B(I+Y)
16Ø B(I+Y)=H
17Ø NEXT Y
18Ø Z = Z - 1
19Ø NEXT I
```

In Zeile 19Ø ist das Sortierprogramm abgeschlossen; es folgt ein Programm für die Ausgabe der sortierten Zahlen.

```
2ØØ FOR I = 1 TO N
21Ø PRINT I, B(I)
22Ø NEXT I
```

Beispiel 2:
Das folgende Programm ist unter Berücksichtigung der „Spartips" zusammengestellt. Es erfüllt die gleiche Aufgabe wie Beispiel 1. Allerdings benötigt es nur 409 Byte und erledigt seinen Job in 20 Sekunden. Es spart damit 40 % Speicherplatz und 20 % Ausführungszeit.

Und so sieht das „Sparprogramm" aus:

```
5DEFINTA–Z:DIMB(1ØØ):CLS:INPUT"WIEVIELE ELEMENTE";N:FORI=1TON:
 INPUT"B=";B(I):NEXT
1ØZ=N–1:FORI =1TON–1:FORY=1TOZ:IFB(I)<B(I+Y)THEN3Ø
2ØH=B(I):B(I)=B(I+Y):B(I+Y)=H
3ØNEXT:Z=Z–1:NEXT
4ØFORI=1TON:PRINTI,B(I):NEXT
```

Den oben aufgezählten Vorteilen stehen jedoch mit bloßem Auge erkennbare Nachteile gegenüber, die ich Ihnen nicht verschweigen will:
- Das Programm ist längst nicht so übersichtlich wie das erste!
- Müssen Sie in dem Programm ändern, dann ist wesentlich mehr „Tipp-Aufwand" erforderlich!
- Vor allem GO...TOs oder IF...THEN erfordern wesentlich mehr Aufmerksamkeit beim Programmieren!

Schlußfolgerung: Auch beim Programmieren hat alles seinen Preis!

13 Fehlersuche

Sie werden die Erfahrung gemacht haben, daß ein leidlich komplexes Programm selten auf Anhieb funktioniert. Das liegt nicht an Ihrem Computer. Sie haben etwas falsch gemacht. Wobei Sie im Grunde drei Möglichkeiten haben, Fehler zu machen:

13.1 Syntax-Fehler

Darunter sind Verstöße gegen die Spielregeln zu verstehen. Häufige Syntax-Fehler können z. B. sein:

- PRINTT... statt PRINT...; hervorgerufen durch Tippfehler oder prellende Tasten.
- A = 3(B+6) anstatt A = 3 * (B + 6)
  Falsches Umsetzen von mathematischen Formeln in BASIC.
- PRINT SIN A statt PRINT SIN (A)
  Falsche Schreibweise von Funktionen; das Argument gehört immer in ( ).
- PRINT MID$(A$(3,1) statt PRINT MID$(A$(3,1))
  Fehler beim Setzen von Klammern.

Stößt der Computer nach RUN im Laufe eines Programms auf einen derartigen Fehler, dann bricht er das Programm ab. Je nach Computer erscheint dann entweder eine Fehler-Nummer – die per Benutzerhandbuch zu der Art des Fehlers führt – oder er „sagt" im Klartext:
? SN ERROR IN 1Ø
Sie wissen dann, daß Sie in Zeile 1Ø gegen die Spielregeln verstoßen haben; Sie können sich mit LIST 1Ø die Zeile ansehen und Ihren Fehler korrigieren. Starten Sie Ihr Programm, so läuft es bis zu einem etwa vorhandenen, weiteren Fehler und steigt erneut aus.

13.2 Ablauf-Fehler

Neben Syntax-Fehlern kann das Programm auch andere Fehler enthalten, auf die der Computer beim zeilenweisen Abarbeiten stößt. Fehler dieser Art sind etwa:
- IF...THEN 15Ø
  In Ihrem Programm gibt es aber keine Zeile 15Ø!

- Sie verwenden in Ihrem Programm 50 indizierte Variablen, haben aber vergessen, am Anfang Ihres Programms durch DIM... entsprechend Speicherplatz zu schaffen.
- Sie haben bei der DIMensionierung von String-Variablen vergessen, mit CLEAR... Platz für Strings zu reservieren!
- Sie haben eine FOR...NEXT...-Schleife nicht mit NEXT abgeschlossen oder mehrere Schleifen dieser Art falsch ineinander geschachtelt.

Sie sehen: die Zahl der Fehlermöglichkeiten ist nahezu unbegrenzt. Auch bei derartigen Fehlern wird der Computer das Programm abbrechen und Ihnen – je nach Gerät – den Fehler entweder in einer Code-Nummer oder mit einem Schlüsselwort ausgeben wie:
...UL ERROR...
...TM ERROR...
...BS ERROR... usw.
In Ihrem Handbuch finden Sie dann, was Code-Nummer oder Kürzel bedeuten; Sie können sich ans Korrigieren machen.

13.3 Logische Fehler

Sind Syntax- und Ablauffehler beseitigt, hat der Computer keinen Grund mehr, sich Ihnen zu verweigern. Er wird das Programm hinter sich bringen und Ihnen ein Ergebnis ausgeben. Aber auch das kann falsch sein! In diesem Falle liegt ein logischer Fehler Ihrerseits vor.

- Sie haben beispielsweise irrtümlich die Formel für die Fläche eines Kreises mit 2*PI*R eingegeben statt mit R ↑ 2 *PI...
- Sie haben Lauf- und/oder Zählvariable nicht richtig gesetzt oder zurückgesetzt...
- Sie haben sich bei IF...THEN... falsch entschieden...
- Ihre Problemanalyse war bereits fehlerhaft und Sie haben daraus die falschen Schlüsse gezogen...

Derartige „logische Fehler" sind deshalb so tückisch, weil Ihnen der Computer ungerührt ein „Ergebnis" liefert; Sie erhalten nicht den geringsten Hinweis darauf, daß dieses Ergebnis falsch sein könnte! Oder erhalten Sie doch einen Hinweis?!

Ist es nicht schon vorgekommen, daß ein hübscher Computerausdruck Ihnen „spanisch" vorkam? Daß Sie dachten: „Das darf doch nicht wahr sein?"...

In diesem Fall haben Sie noch Glück gehabt und können sich ans Kontrollieren machen. Wenn Sie aber arglos sind und keinen Fehler vermuten..., dann ist guter Rat teuer!

Es gibt keine Tips dafür, wie man logische Fehler findet. Aber ich werde Ihnen einige Tips geben, wie man logische Fehler vermeidet:

- Halten Sie sich an die im Abschnitt 3 beschriebenen Hinweise für den Aufbau eines Programms!
- Lassen Sie sich während der Programmentwicklung Zwischenrechnungen machen und prüfen Sie diese auf Richtigkeit!
- Arbeiten Sie dabei mit kleinen, „glatten" Zahlen, die Sie zur Not noch „im Kopf" verfolgen können.
- Versuchen Sie, Ihre Programme in Blocks zu zerlegen, die Sie zunächst einzeln testen und bei Fehlerfreiheit aneinanderhängen!
- Da man das Rad nicht ständig neu erfinden sollte, ist es durchaus vernünftig, in ein Programm Teile oder Routinen einzubauen, die sich bereits in anderen Programmen als richtig erwiesen haben!

Aber schauen Sie sich derartige Bausteine vorher sorgfältig an!
- Stimmen die Variablennamen mit denen Ihres Programms überein?
- Stimmen etwaige Grenzen mit denen Ihres Programms überein?
- Müssen Sie neu DIMensionieren? usw. usw.

Zum Schluß noch zwei BASIC-Kommandos, die bei der Fehlersuche in bescheidenem Maße Hilfestellung leisten können; vor allem dann, wenn Sie IF...THEN...-Verzweigungen nachspüren wollen.

Es handelt sich um das TRACE-Kommando[1]), das Ihnen auf dem Schirm neben dem Ergebnis auch die abgearbeiteten Zeilen-Nummern anzeigt. Üben Sie das an einem kleinen Beispiel:

```
5 CLS
1Ø A = 1
2Ø PRINT A
3Ø A = A + 1
4Ø IF A = 1Ø THEN 6Ø
5Ø GOTO 2Ø
6Ø END
```

Geben Sie jetzt TRON[1]) ein, danach ENTER und – wie üblich – RUN, gefolgt von ENTER. Der TRS-80 gibt beispielsweise aus:

---

[1]) Anmerkung:

	TRS-80	PC 100	APPLE II	PET	TI 99/4
TRACE-Funktion EIN	TRON	–	TRACE	–	TRACE
TRACE-Funktion AUS	TROFF	–	NOTRACE	–	UNTRACE

```
>1Ø<>2Ø< 1
>3Ø<>5Ø<>2Ø< 2
>3Ø<>5Ø<>2Ø< 3
>3Ø<>5Ø<>2Ø< 4
>3Ø<>5Ø<>2Ø< 5
>3Ø<>5Ø<>2Ø< 6
>3Ø<>5Ø<>2Ø< 7
>3Ø<>5Ø<>2Ø< 8
>3Ø<>5Ø<>2Ø< 9
>3Ø<>4Ø<>6Ø<
```

Mit TROFF[1]) können Sie TRACE wieder abschalten.

14 Programm-Bibliothek

Ich denke, das meiste über BASIC haben Sie inzwischen kennengelernt. Es gilt nun, diese Kenntnisse solange einzuüben, bis Sie Ihnen „in Fleisch und Blut" übergegangen sind.

Dazu soll die folgende Programm-Bibliothek dienen. Die darin aufgenommenen Programme decken einen möglichst breiten Bereich sinnvollen Computereinsatzes ab.

Es kommt mir weniger darauf an, Ihnen sofort einsetzbare, fertige Kochrezepte zu liefern. Vielmehr möchte ich mich bemühen, Sie zunächst an das jeweilige Problem heranzuführen, dieses zu durchleuchten und damit einer Programmierung mit Ihrem frisch erworbenen BASIC-Wissen zugänglich zu machen.

Ich hoffe, daß dabei Ihre Phantasie angeregt wird; daß Sie aus den vorgestellten eigene Programme entwickeln oder nach Ihren Vorstellungen abwandeln. Beginnen wir:

14.1 Nützliche Programm-Routinen

14.1.1 Ausstieg aus Schleifen

Das Arbeiten mit Zählschleifen ist Ihnen längst aus zahlreichen Programmen geläufig.

```
10 A = 12
20 PRINT A,A/3
30 A = A + 1
```

Kopfzerbrechen kann nun die nächste Zeile machen, in der Sie üblicherweise unterbringen, wie lange Sie dieses Spiel treiben wollen. Nehmen Sie an, Sie wollen in dem obigen Beispiel „aussteigen", wenn A = 24 ist. Fahren Sie fort mit:

```
40 IF A = 24 THEN 60
50 GOTO 20
60 END
```

Nun müssen Sie sich erinnern, daß die PRINT-Zeile für A = 24 nicht mehr ausgeführt wird, weil das Programm zu 60 END springt, sobald in 40 A = 24 erkannt ist.

Dieser „Ausstieg" ist also nur sinnvoll, wenn Sie die Zeile mit A = 24 besonders herausheben wollen. Wie etwa in:

```
60 PRINT" DIE LETZTE RECHNUNG ERGIBT :";A,A/3
65 END
```

Normalerweise ist es besser, eine andere „Austrittsformel" zu wählen. Wie wäre es mit – zunächst verbal –
– "...Rechne solange, bis A größer ist als 24..."

Das ergibt dann:

```
40 IF A > 24 THEN 60
50 GOTO 20
60 END
```

Verfügt Ihr Computer über IF...THEN...ELSE..., dann geht's ganz kurz und elegant:

```
40 IF A > 24 THEN 60 ELSE 20   oder:
40 IF A > 24 THEN END ELSE 20
```

Bei Bedarf können Sie eine Zählschleife auch „rückwärts" oder „abwärts" laufen lassen!

```
10 A = 10
20 PRINT A
30 A = A - 1
40 IF A < 1 THEN 60
50 GOTO 20
60 END
```

Schwieriger wird es erst, wenn Sie Zeile 30 nicht um 1 bzw. um Werte erhöhen oder erniedrigen, deren Verlauf Sie noch im Kopf übersehen können. Wo wollen Sie beispielsweise in dem folgenden Beispiel aussteigen?

```
10 A = 1.25
20 PRINT A
30 A = A + .73
```

Wenn Sie in einem solchen Falle wenigstens wissen, *wie oft* Sie die Rechnung durchführen lassen wollen, dann können Sie einen entsprechenden Zähler mitlaufen lassen.

Soll die Schleife 20mal durchlaufen werden, ergänzen Sie das letzte Programm um:

```
15 I = 1
35 I = I + 1
40 IF I>20 THEN 60
50 GOTO 20
60 END
```

Wie Sie bemerken, werden I und sein Verlauf auf dem Schirm überhaupt nicht sichtbar. Möchten Sie I als „Laufnummer" mit abbilden, so heißt es:

```
20 PRINT I,A
```

Bei abwärts zählenden Schleifen wollen Sie gelegentlich das Spiel dann abbrechen, wenn A „durch 0 geht". Hier bietet sich für den Ausstieg an:

```
40 IF SGN(A) =-1... oder
40 IF SGN(A) = 0...
```

14.1.2 Warteschleifen

Wollen Sie den Ablauf des Programms eine (kurze) Zeit anhalten, etwa um ein Zwischenergebnis zu betrachten oder um dem Benutzer eine Mitteilung zu machen, können Sie die Warteschleife verwenden:

```
...FOR X = 10 TO 1000:NEXT
```

Diese Zwangspause wird um so länger, je größer Sie die Zahl nach X machen. 1000 dauert ca. 1,5 Sekunden; Zeit, um etwa eine halbe Bildschirmzeile zu lesen und zu „verdauen".

14.1.3 Aufhalten des Programms mit INPUT

Wie kurz oder wie lang Sie auch die Warteschleife wählen, Sie üben damit einen Zwang auf den Benutzer Ihres Programmes aus. Sie sollten daran denken, daß Ihr Programm auch von Fremden – unter Umständen von Computer-Laien – verwendet werden kann. Und da ist es schon „benutzerfreundlicher", wenn Sie dem Benutzer überlassen, wann er weitermachen will.

Das machen Sie so:

```
TEXT MIT ERKLAERUNGEN FUER DEN BENUTZER

300 PRINT" WENN SIE BEREIT SIND, DRUECKEN SIE DIE TASTE 'ENTER' "
310 INPUT Q
```

Nun wartet der Computer. Er interpretiert den Druck auf ENTER als Q = 0 und setzt das Programm fort. Vorausgesetzt, Q ist nicht der Name einer in Ihrem Programm benutzten Variablen, schadet das nicht weiter! Jedenfalls kann sich der Benutzer jetzt beliebig viel Zeit lassen!

14.1.4 Menütechnik mit ON...GOTO...

Da wir gerade bei „benutzerfreundlich" sind: verwenden Sie am Anfang Ihres Programms ausreichend Erklärungen, um dem Benutzer zu sagen:
– was das Programm tut
– und was es (das Programm) an Mitarbeit vom Benutzer erwartet.
Ich stelle mir das etwa so vor:

DIESES PROGRAMM BERECHNET IHRE
ABSCHREIBUNGEN
WENN SIE SEHEN WOLLEN, WIE SIE
DAS PROGRAMM NUTZEN KOENNEN, DANN
DRUECKEN SIE DIE TASTE ‚ENTER'

Machen Sie sich dafür ruhig ein wenig Mühe; Nutzen Sie PRINT TAB... oder PRINT AT, um den Schriftblock auch optisch gefällig auf den Schirm zu bringen. Je appetitlicher das aussieht, um so lieber arbeitet der Benutzer damit.
Und dann wieder:

```
100 INPUT Q
```

Und nun weiter:

AUSWAHLTABELLE
LINEARE ABSCHREIBUNG
....KENNZIFFER 1
DEGRESSIVE ABSCHEIBUNG
....KENNZIFFER 2
DRUECKEN SIE DIE GEWUENSCHTE
KENNZIFFER, DANACH ‚ENTER'

```
200 INPUT Q
300 ON Q GOTO 400,500
```

Nun beginnen Sie bei 400 Ihr Programm für die lineare, bei 500 Ihr Programm für die degressive Abschreibung. Beginnen Sie jeweils wieder mit einem Text. Etwa, welche Eingaben Sie für die Berechnung erwarten usw.

Haben Sie das Ergebnis präsentiert, sollten Sie den Benutzer fragen,
– ob er noch eine Berechnung haben möchte? Dann gehen Sie zurück in die Auswahltabelle.
– andernfalls sollte sich Ihr Computer vom Benutzer verabschieden. Und Feierabend machen.

Zusammenfassend:
Ein gutes Programm sollte in folgenden Blocks aufgebaut sein:

- Das Programm stellt sich vor. Was tut es? Was kann es?
- Das Programm erklärt, wie man es bedient. Es enthält eine Auswahltabelle – „Menü" – bei mehrteiligen Programmen.
- Abfrage des Benutzers nach den für die Berechnung nötigen Werten.
- Präsentation der Ergebnisse.
- Frage nach dem weiteren Fortgang. Ggf. Neubeginn oder Verabschiedung.

Ich habe versucht, Ihnen das in einigen Übungsprogrammen nahezubringen – wenn auch aus Platzgründen nicht bei allen. Je öfter ein Programm genutzt wird, um so eher lohnt sich der Aufwand für derartige „Schnörkel".

14.1.5 Sperren gegen Falscheingaben

Selbst Ihnen wird es gelegentlich passieren, daß Sie bei der Eingabe eine „falsche" Taste drücken. Bei der Benutzung Ihres Programms durch Ungeübte müssen Sie natürlich damit ganz besonders rechnen. Und nur ein sehr benutzerunfreundliches Programm läuft bei einer Fehleingabe ungerührt an Anfang oder Ende des Programms. Bauen Sie eine Sperre gegen nicht zugelassene Eingaben ein und machen Sie den Benutzer auf seinen Fehler aufmerksam!

Eine derartige Routine bauen Sie in Ihr Programm ein, unmittelbar bevor Sie mit ON..GOTO.. zu den einzelnen Programmblocks übergehen. Das folgende Beispiel unterstellt, daß der Benutzer drei Teilprogramme durch Eingabe der Ziffern 1, 2 oder 3 abrufen kann.

```
290 IF Q < 1 OR Q > 3 THEN 1000
300 ON Q GOTO 400,600,800
400 .....Beginn Programm 1
600 .....Beginn Programm 2
800 .....Beginn Programm 3

1000 PRINT" FALSCHEINGABE!"
1010 PRINT"BITTE TASTE 1,2 ODER 3 DRUECKEN!"
1020 FOR X = 1 TO 1000 : NEXT : CLS : GOTO....zur Auswahltabelle!
```

Zeile 290 prüft also die Eingabe auf Zulässigkeit. Im Fehlerfalle erhält der Benutzer mit 1000 die Mitteilung, daß er sich geirrt hat. Nach einer schicklichen Pause (1020) kehrt das Programm automatisch zur Auswahltabelle zurück.

14.1.6 Blockweise Ausgabe von Ergebnissen

Ziel aller Mühen ist es, am Ende des Programms vom Computer eine Ausgabe zu erhalten. Besteht diese etwa aus einer Tabelle, dann kann es Probleme geben, diese ganz auf den Bildschirm zu bekommen.

Üblicherweise faßt der Bildschirm 16 Zeilen. Davon mögen zwei für eine Überschrift draufgehen; zwei Zeilen sind ferner für die READY-Meldung am Schluß der Ausgabe zu berücksichtigen. Hat die Ausgabe mehr als zehn Zeilen, dann werden die Überschrift und u. U. einige der ersten Zeilen nach oben „weggerollt", sind also für den Benutzer nicht mehr sichtbar. Mit dem folgenden Vorschlag können Sie die Ausgabe nun so gestalten, daß immer nur Blocks à zehn Zeilen serviert werden.

Es wird bei dem Vorschlag unterstellt, daß „vorn" in Ihrem Programm die Anzahl der auszugebenden Zeilen bekannt ist (N). Etwa durch die Zeile:
FOR I = 1 TO N...

Sie programmieren dann Ihren Ausgabeteil wie folgt:

```
990 A = 1 : B = 10
```

Damit meinen Sie die ersten zehn Zeilen.

```
 992 GOTO 1000
 995 A = A + 10 : B = B + 10
 997 IF B > N THEN 2000

1000 CLS : FOR X = A TO B

1020 PRINT......Es folgt Ihre Ausgabe!
1030 NEXT X
1040 INPUT Q
```

Damit ist der erste Block mit zehn Zeilen auf dem Schirm. Durch Zeile 1040 haben Sie beliebig Zeit, die Ausgabe zu studieren. Wollen Sie fortfahren, drücken Sie ENTER, und es geht mit den Ausgabezeilen 11 bis 20 wie folgt weiter:

```
1050 CLS : GOTO 995
```

Und so setzt sich der Ablauf fort. Nehmen Sie an, Ihr Programm würde insgesamt 35 Ausgabezeilen liefern. Dann wird der Computer bei der „4. Seite" (Zeilen 31...40) erkennen, daß Zeile 997 des Programms wahr ist: B – inzwischen 40 – ist größer als N (hier: 35!).

Der Computer muß also nur noch die verbleibenden Zeilen 3Ø bis 35 ausgeben. Das tut er wie folgt:

```
2ØØØ FOR X = A TO N
2Ø1Ø PRINT.....Wiederholung Ihrer Ausgabe wie 1Ø2Ø
2Ø2Ø NEXT X
```

Und nun setzen Sie noch einen krönenden Abschluß:

```
2Ø3Ø PRINT"ENDE DER AUSGABE VON INSGESAMT";N;" ZEILEN"
```

14.1.7 Bildung von Summen, Durchschnittswerten, „N-Quadrat"

Oft kommt in einem Programm die Detailaufgabe vor, aus N eingegebenen Größen
- die Summe aller Größen zu bilden...
- den Durchschnittswert zu ermitteln. Darunter ist hier zu verstehen: Summe aller Größen geteilt durch Anzahl N
- die Summe der Quadrate aller Größen zu bilden. Also: $(A1)^2 + (A2)^2 + (A3)^2 + ...(AN)^2$

Die grundsätzliche Vorgehensweise bei Aufgaben dieser Art soll an dem folgenden Beispiel erläutert werden. Zunächst wird für die Übung ein Eingabeteil entworfen:

```
 5 CLS
1Ø N = 1Ø
```

Für die Summe wählen Sie den Namen S, für den Durchschnitt den Namen D und für die Summe der Quadrate den Namen Q. Alle sind natürlich zu Beginn Ø.

```
2Ø S = Ø : Q = Ø
3Ø FOR I = 1 TO N
4Ø INPUT" ZAHL EINGEBEN !";A(I)
5Ø S = S + A(I)
```

Mit Zeile 5Ø bilden Sie die Summe!

```
6Ø Q = Q + (A(I) ↑ 2)
7Ø NEXT I
```

Und wo bleibt D? Schauen Sie gleich auf Zeile 11Ø!

Für die Ausgabe gilt:

```
100 PRINT"SUMME ALLER A=";S
110 PRINT"DURCHSCHNITTSWERT D =";S/N
120 PRINT"SUMME DER A-QUADRAT =";Q
```

Und nun lassen Sie das einmal „laufen"! Wählen Sie für A „einfache" Zahlen, damit Sie das Ergebnis leicht auf Richtigkeit überprüfen können.

14.1.8 Rundungen

Üblicherweise rechnen Personal-Computer ohne besondere Maßnahmen mit „einfacher Genauigkeit"; d. h., sie geben das Ergebnis mit sechs signifikanten Stellen aus.

123456
1.23456
123.456 usw.

Wollen Sie in den letzten Beispielen nicht drei oder fünf Stellen hinter dem Komma, so lassen Sie Ihren Computer auf die gewünschte Stellenzahl „runden".

Wenn es in Ihren Rechnungen um Geld geht, sind Sie meist mit zwei Stellen hinter dem Komma zufrieden. Dafür gilt die Rundungsformel:

```
...A = INT(A*100 + .5)/100
```

Wollen Sie A auf drei oder vier Stellen – allgemein: auf N Stellen – runden, wählen Sie die Formel:

```
...A = INT(A*10↑N + .5)/10↑N
```

14.1.9 Vereinzeln

Darunter soll hier folgende Aufgabenstellung verstanden werden: Sie wollen die Variable A = 123 in „Unter"-Variable zerlegen; und zwar soll:

H die „Hunderter" darstellen, also den Wert 1 erhalten
Z die „Zehner" darstellen, also den Wert 2 erhalten
E die „Einer" darstellen, also den Wert 3 erhalten.

Das können Sie zunächst mit Hilfe der INT..-Funktion und etwas Überlegung anstellen.

Beispiel: Die Variable A = 3.5 soll zerlegt werden in:

```
E = 3
Z1 = 5
```

Das geht so:

```
E = INT(A)
Z1 = A*1Ø - INT(A)*1Ø
```

Bei anderen Größen für A verfahren Sie sinngemäß; ggf. müssen Sie bei der Zerlegung Hilfvariable einführen.

Eine weitere Möglichkeit, derartige Aufgaben zu lösen, besteht darin, A zunächst in einen String zu verwandeln, um daraus mit Hilfe der verschiedenen String-Funktionen die einzelnen Stellen zu isolieren.

Wählen Sie als Beispiel:

```
1Ø A = 351.1
```

Die Umwandlung von A in einen String geschieht mit:

```
2Ø A$ = STR$(A)
```

Die zu gewinnende „Hilfsvariable" H (für Hunderter) ist nun das zweite (!) Zeichen von A$. Weil der Computer bei der Umwandlung ein Zeichen als Blank für das Vorzeichen freihält. Das wird aber hier – weil A positiv ist – nicht sichtbar.

```
3Ø H$ = LEFT$(A$,2)
```

Da Sie nur an dem Zahlenwert interessiert sind, fahren Sie fort:

```
35 H = VAL(H$)
```

Für die Zehner gilt:

```
4Ø Z$ = MID$(A$,3,1)
45 Z = VAL(Z$)
```

Sind Sie bereits in der „String-Manipulation" geübt, können Sie die Zeilen 3Ø,35 und 4Ø,45 zusammenfassen, wie das in den folgenden Zeilen 5Ø und 6Ø vorexerziert wird:

```
5Ø E = VAL(MID$(A$,4,1))
```

Da das Komma die fünfte Position hat, müssen Sie für die Zehntel schreiben:

```
6Ø Z1 = VAL(MID$(A$,6,1))
```

Damit Sie sehen, ob das funktioniert hat, lassen Sie sich zeigen:

```
7Ø PRINT H,Z,E,Z1
```

Das Ergebnis muß sein:

```
3          5          1          1
```

Aus A = 351.1 haben Sie also vier neue Variable gewonnen, mit denen Sie nach Herzenslust weiterarbeiten können.

14.1.10 Erzeugen von Zufallszahlen

Alle Personal-Computer verfügen über die Fähigkeit, Zufallszahlen ausgeben zu können; sie tun das mit Hilfe der Funktion RND(X).

Die Art und Weise, wie die Funktion ausgegeben wird, ist jedoch bei den einzelnen Typen verschieden, so daß ein Blick in das jeweilige Handbuch ratsam ist.

Die folgenden Ausführungen gelten für die Maschinen:

TRS-80	PC100/AIM 65
PRINT RND(Ø)	PRINT RND(Ø)
Der Computer gibt bei jeder Wiederholung eine andere Zahl aus, die im Bereich Ø...1 liegt.	Der Computer gibt bei jeder Wiederholung die gleiche Zahl im Bereich Ø...1 aus.

Wollen Sie etwa eine Zufallszahl zwischen Ø und 5Ø, so heißt es:

PRINT RND(5Ø)	PRINT INT(5Ø*RND(1)+1)

Wollen Sie zehn Zufallszahlen zwischen 1 und 5Ø, so heißt es:

1Ø FOR I = 1 TO 1Ø	1Ø FOR I = 1 TO 1Ø
2Ø PRINT RND(5Ø) : NEXT	2Ø PRINT INT(5Ø*RND(1)+1) : NEXT

14.1.11 Sie suchen aus einer Reihe von Zahlen die kleinste

Stellen Sie sich vor, Sie haben in einer DATA-Liste zehn Zahlen, aus der Sie die kleinste heraussuchen wollen. Sie können dann die Zeilen 1Ø...11Ø des gleich

folgenden Programms für diese Aufgabe verwenden. (Die Daten stehen in Zeile 3ØØ!) In diesem Programm findet die Suche nach der kleinsten Zahl in den Zeilen 6Ø...1ØØ statt.

14.1.12 Sie suchen aus einer Reihe von Zahlen die größte

Für diese Aufgabe gelten die Zeilen 16Ø...24Ø des gleichen Programms, wobei die Suche nach der größten Zahl in den Zeilen 2ØØ...24Ø erfolgt.

14.1.13 Sie suchen aus einer Reihe von Zahlen die größte und die kleinste

Dazu können Sie schließlich das gesamte Programm, Zeilen 1Ø...3ØØ verwenden.

```
 1Ø DIM (5Ø)
 2Ø READ N
 3Ø FOR J = 1 TO N
 4Ø READ X (J)
 5Ø NEXT J
 6Ø K = X (1)
 7Ø FOR I = 2 TO N
 8Ø IF X (I) > K THEN 1ØØ
 9Ø K = X (I)
1ØØ NEXT I
11Ø PRINT "DIE KLEINSTE ZAHL IST";K
15Ø RESTORE
16Ø READ N
17Ø FOR J = 1 TO N
18Ø READ X (J)
19Ø NEXT J
2ØØ K = X (N)
21Ø FOR I = 2 TO N
22Ø IF X (I) < K THEN 24Ø
23Ø K = X (I)
24Ø NEXT I
25Ø PRINT "DIE GROESSTE ZAHL IST"; K

3ØØ DATA 1Ø, 34, 23, 35, 99, 81, 72, 27, 29, 38, 9Ø
```

Das Programm hat eine Besonderheit, auf die ich Sie aufmerksam machen möchte, weil darin wieder eine Routine steckt, die Sie oft verwenden können.

Zeile 2Ø heißt schlicht ...READ N. Damit liest der Computer das erste Datum in der DATA-Liste in Zeile 3ØØ (im Beispiel: 1Ø!) und setzt dieses als „Anzahl der

Daten" in Zeile 30 in die FOR...NEXT...-Schleife ein. Haben Sie also in der Liste mehr als 10 Daten, müssen Sie das erste Datum gleich der Gesamtzahl Ihrer Daten wählen.

Die Suche nach der größten und der kleinsten Zahl können Sie natürlich auch über ein Sortierprogramm lösen. Denn nach dem Sortieren einer Zahlenliste ist doch:

– die zuerst ausgegebene Zahl die kleinste...
– die zuletzt ausgegebene Zahl die größte Zahl.

Das ist im folgenden Beispiel dargestellt, in dem es mir nur auf den Ausgabeteil – nicht den Sortierteil – ankommt. Das Sortieren geschieht in den Zeilen 100 bis 190. Die Zahlen liegen nicht in einer DATA-Liste vor, sondern werden über INPUT... eingegeben. Schließlich ist mit 20, 50, 250, 260 und 320 auch noch der Durchschnittswert errechnet.

```
  5 CLS
 10 N = 8
 20 S = 0
 30 FOR I = 1 TO N
 40 INPUT "MESSWERT EINGEBEN!";C (I)
 50 S = S + C (I)
 60 NEXT I

100 Z = N - 1
110 FOR I = 1 TO N - 1
120 FOR Y = 1 TO Z
130 IF C (I) < C (I+Y) THEN 170
140 H = C (I)
150 C (I) = C (I+Y)
160 C (I+Y) = H
170 NEXT Y
180 Z = Z - 1
190 NEXT I

250 D = S/N
260 D = INT (D*10 + .5) / 10

295 CLS
300 PRINT "DER MINIMALWERT IST ......... "; C (1)
310 PRINT "DER MAXIMALWERT IST ........ ";C (N)
320 PRINT "DER DURCHSCHNITTWERT IST ... "; D
```

14.1.14 Allgemeines zu Sortierprogrammen

Das Sortieren von Daten zählt in der kommerziellen Datenverarbeitung zu den Aufgaben, die vom Computer am häufigsten zu bewältigen sind. Es ist deshalb naheliegend, daß es dafür eine große Zahl von Programmen gibt, die sich im wesentlichen durch die Zeit unterscheiden, die der Computer zu ihrer Abarbeitung benötigt.

In dem vorigen Abschnitt hatten Sie in den Zeilen 1ØØ...19Ø bereits ein Sortierprogramm kennengelernt. Es ist vergleichsweise langsam, dafür ist es leicht zu durchschauen[1]) und erfordert nur wenige Programmzeilen.

Die folgenden Sortierprogramme sind in der Literatur unter dem Namen „Quick-Sort" bekannt. Wie der Name vermuten läßt, arbeiten sie schnell. Leider erfordern sie aber auch einen höheren Programmieraufwand. Die Programme sind jeweils in drei Teile geteilt:

- Eingabeteil
- Sortierteil
- Ausgabeteil. Dieser wird in zwei Versionen geliefert:
  - Ausgabe aufsteigend von 1...N bzw. von A...Z,
  - Ausgabe absteigend von N...1 bzw. von Z...A

  Beachten Sie, daß auf- oder absteigendes Sortieren hier also eine Frage der Ausgabe ist.

14.1.15 Sie sortieren Zahlen

EINGABE

```
  4 DIM A (1ØØ)
  5 CLS
 1Ø INPUT "WIEVIELE ZAHLEN"; N
 2Ø FOR I = 1 TO N
 3Ø INPUT "ZAHL ="; A (I)
 4Ø NEXT I
```

```
12Ø L = 1
13Ø B (L) = N+1
14Ø M = 1
15Ø J = B (L)
```

[1]) Eine genaue Erklärung des Sortierprogramms finden Sie in: Rudolf Busch „BASIC für Einsteiger, Franzis-Verlag 1982.

SORTIER-ROUTINE („Quick-Sort")		160 I = M−1 170 IF J−M < 3 THEN 330 180 M1 = INT ((I+J) /2) 190 I = I+1 200 IF I = J THEN 270 210 IF A (I) < = A (M1) THEN 190 220 J = J−1 230 IF I = J THEN 270 240 IF A (J) > = A (M1) THEN 220 250 A (0) = A (I):A (I) = A (J):A (J) =A (0) 260 GOTO 190 270 IF I > = M1 THEN I = I−1 280 IF J = M1 THEN 300 290 A (0) = A (I):A (I) = A (M1):A (M1) = A (0) 300 L = L+1 310 B (L) = I 320 GOTO 150 330 IF J − M < 2 THEN 360 340 IF A (M) < A (M+1) THEN 360 350 A (0) = A (M):A (M) = A (M+1):A (M+1) = A (0) 360 M = B (L) +1 370 L = L−1 380 IF L > 0 THEN 150
AUSGABE	b) nach steigenden Zahlen	400 FOR I = 1 TO N 410 PRINT A (I) 420 NEXT I
	a) nach fallenden Zahlen	400 FOR I = N TO 1 STEP − 1 410 PRINT A (I) 420 NEXT I

14.1.16 Sie sortieren Strings

EINGABE

```
  2 CLEAR 1000
  4 DIM A$ (100)
  5 CLS
 10 INPUT "WIEVIELE NAMEN"; N
 20 FOR I = 1 TO N
 30 INPUT "NAME ="; A$ (I)
 40 NEXT I
```

SORTIER-ROUTINE („Quick-Sort")

```
120 L = 1
130 B (L) = N+1
140 M = 1
150 J = B (L)
160 I = M−1
170 IF J−M < 3 THEN 330
180 M1 = INT ((I+J) /2)
190 I = I+1
200 IF I = J THEN 270
210 IF A$ (I) < = A$ (M1) THEN 190
220 J = J−1
230 IF I = J THEN 270
240 IF A$ (J) > = A$ (M1) THEN 220
250 A$ (0) = A$ (I):A$ (I) = A$ (J):A$ (J) = A$ (0)
260 GOTO 190
270 IF I > = M1 THEN I = I−1
280 IF J = M1 THEN 300
290 A$ (0) = A$ (I):A$ (I) = A$ (M1):A$ (M1) = A$ (0)
300 L = L+1
310 B (L) = I
320 GOTO 150
330 IF J − M < 2 THEN 360
340 IF A$ (M) < A$ (M+1) THEN 360
350 A$ (0) = A$ (M):A$ (M) = A$ (M+1):A$ (M+1) = A$ (0)
360 M = B (L) +1
370 L = L−1
380 IF L > 0 THEN 150
```

AUSGABE	b) nach A...bis Z...	400 FOR I = 1 TO N 410 PRINT A$ (I) 420 NEXT I
	a) nach Z...bis A...	400 FOR I = N TO 1 STEP – 1 410 PRINT A$ (I) 420 NEXT I

14.1.17 Das Sortieren wird komplizierter

Nehmen Sie an, eine Liste von Namen, die Sie sortieren wollen, beginnt mit:

A$	B$	C$
Napp, Werner	Steinweg 7	1 Berlin 17
Weber, Horst	Waldstraße 5	8 München 1
Abel, Klaus	Zinnstraße 6	2 Hamburg 1
... usw.		

Nehmen Sie ferner an, daß Name, Straße und Ort je in einem eigenen String stecken. Sortieren Sie das nach A$ mit dem Programm von vorhin, dann erhalten Sie:

ABEL,KLAUS	STEINWEG 7	1 BERLIN 17
NAPP,WERNER	WALDSTRASSE 5	8 MUENCHEN
WEBER,HORST	ZINNSTRASSE 6	2 HAMBURG 1

Wie Sie sehen, sind zwar die Namen – die A$ – richtig sortiert. Leider sind die Herren jedoch gezwungen worden, umzuziehen...

Offensichtlich ist es notwendig, zwar nach A$ zu sortieren. Jedoch müssen in etwaigen „Vertauschungszeilen" des Programms zu den A$ jeweils die B$ und C$ *mit umgestellt* werden.

Die Lösung des Problems ist einfacher, als Sie glauben. Zur Erklärung verwende ich das einfacher zu durchschauende Sortierprogramm von Seite 137. Dieses heißt zunächst – auf Strings abgewandelt:

```
100 Z = N - 1
110 FOR I = 1 TO N - 1
120 FOR Y = 1 TO Z
130 IF A$(I) < A$(I+Y) THEN 170
```

```
140 H$ = A$(I)
150 A$(I) = A$(I+Y)
160 A$(I+Y) = H$
```

```
170 NEXT Y
180 Z = Z - 1
190 NEXT Z
```

Oben ist bereits eingerahmt, worauf es ankommt: in den Zeilen 140...160 findet das Vertauschen der Variablen statt für den Fall, daß Zeile 130 NICHT WAHR ist. In unserem Beispiel mit den Namen von Napp bis Abel ändern sich nun lediglich die Zeilen 140...160 wie folgt:

```
140 H$=A$(I):H1$=B$(I):H2$=C$(I)
150 A$(I)=A$(I+Y):B$(I)=B$(I+Y):C$(I)=C$(I+Y)
160 A$(I+Y)=H$:B$(I+Y)=H1$:C$(I+Y)=H2$
```

So einfach ist das!

Nun sollten Sie dieser Segnungen auch bei „Quick-Sort" teilhaftig werden. Nehmen Sie sich noch einmal das Programm zum Sortieren von Strings vor! Auch darin gibt es die erwähnten „Vertauschungen". Allerdings kommen diese bei Quick-Sort gleich dreimal vor; nämlich in den Zeilen 250, 290 und 350!

Alle drei Zeilen müssen nach dem obigen Schema umgestellt werden! Es ist dabei jeweils sorgfältig auf die unterschiedlichen Indices in ( ) zu achten!

Nur für die Zeile 250 mache ich Ihnen das vor; den Rest können Sie wohl bei Bedarf selbst besorgen?

```
250 A$(0)=A$(I):B$(0)=B$(I):C$(0)=C$(I):A$(I)=A$(J):B$(I)=B$(J):C$(I)=C$(J):
    A$(J)=A$(0):B$(J)=B$(0):C$(J)=C$(0)
```

Wenn Ihr Computer das Monstrum in Zeile 250 nicht in einer Zeile akzeptiert, dann müssen Sie schreiben:

```
250 A$(0)=A$(I):B$(0)=B$(I);C$(0)=C$(I)
252 A$(I)=A$(J):B$(I)=B$(J):C$(I)=C$(J)
254 A$(J)=A$(0):B$(J)=B$(0):C$(J)=C$(0)
```

Und nun können Sie wohl nachfühlen, warum Sortierprogramme so lange dauern!?

14.1.18 Der Computer ermittelt automatisch die Anzahl der Daten

Arbeiten Sie in Programmen mit INPUT... oder READ...DATA..., dann müssen Sie dem Computer zu Anfang des Programms mitteilen, mit wieviel Daten/Eingaben er es zu tun bekommen wird. Das kann dann lästig werden, wenn Sie zunächst selbst noch nicht wissen, wieviele Daten/Eingaben zusammenkommen werden. Lassen Sie den Computer das selbst herausfinden!

● Ermittlung der eingegebenen Datenzahl bei INPUT...

Als Vorbereitung stellen Sie genügend Platz für Strings bereit und dimensionieren nach dem geschätzten Bedarf:

```
5 CLEAR 1000
10 DIM A$(100)
```

Dann fahren Sie fort:

```
50 FOR I = 1 TO 100
60 PRINT"GEBEN SIE DEN STRING EIN !"
70 INPUT A$(I)
```

Ihr Computer wird nun nacheinander Ihre Eingabe von insgesamt 100 Strings entsprechend Ihrer Dimensionierung in Zeile 10 bzw. der Länge der Schleife in Zeile 50 abfragen.
Nehmen Sie an, Sie haben aber nur 37 Strings. Dann geben Sie als 38. einfach „ENDE" ein! Vorher setzen Sie ihr Programm fort:

```
80 IF A$(I) = "ENDE" THEN N = I - 1 : GOTO 500 : NEXT I
```

Jetzt wird Ihr Computer wissen, daß Sie insgesamt 37 Strings eingegeben haben und für den Rest des Programms – ab Zeile 500 – damit operieren.

Natürlich können Sie diese Technik auch für die Eingabe von numerischen Variablen verwenden. Sie geben dann als letzte Variable 9999 ein und ändern Zeile 80 zu:

```
80 IF A(I) = 9999 THEN N = I - 1...
```

Übrigens: warum N = I – 1?
Nun, „ENDE" oder 9999 war jeweils (im Beispiel) die 37 + 1 te Eingabe!

● **Ermittlung der eingegebenen Datenzahl bei READ...DATA...**

Hier können Sie in den READ...DATA...-Teil des Programms eine Zählschleife zum „Daten-Zählen" einbauen.

```
10 N = 0
20 READ A$
30 N = N + 1
40 IF A$ = "ENDE" THEN 60
50 GOTO 20
60 I = N - 1
```

Damit haben Sie die Anzahl der Daten ermittelt. Bevor Sie mit Zeile 70 in Ihrem Programm fortfahren, NICHT VERGESSEN:

```
65 RESTORE
70 FOR X = 1 TO I
80 READ A$(X)
90 PRINT A$(X)
100 NEXT X
```

Auch hier können Sie mit numerischen Variablen sinngemäß verfahren:

```
20 READ A
30...
40 IF A = 999 THEN...
```

14.1.19 Sie lassen den Computer selbständig DIMensionieren

Sie arbeiten mit indizierten Variablen, die – für das Beispiel – in drei Spalten angeordnet sind; die Anzahl N der Zeilen sei Ihnen zunächst noch unbekannt.

Hier können Sie folgendes Schema wählen:

```
5 S = 0
10 READ A
15 S = S + 1
20 IF A = 0 THEN 30
25 GO TO 10
30 N = (S - 1)/3
```

Sie lassen nun den Computer DIMensionieren:

```
35 RESTORE
40 DIM A (N,3)        das bedeutet hier: N Zeilen zu 3 Spalten!
45 FOR X = 1 TO N
50 FOR B = 1 TO 3
55 READ A(X,B)
60 NEXT B,X
```

Schreiben Sie zur Übung folgende Datenliste:

```
200 DATA 1,2,3,4,5,6,7,8,9,0
```

Lassen Sie sich zur Kontrolle zeigen:

```
100 FOR X = 1 TO N
110 PRINT A(X,1),A(X,2),A(X,3)
120 NEXT X
```

Das Ergebnis wird sein:

```
1        2        3
4        5        6
7        8        9
```

14.2 Programme aus Mathematik und Geometrie

14.2.1 Sie lösen Gleichungen mit zwei Unbekannten

In der Gleichung: 2X + 3 Y = 13 sind die Unbekannten X und Y.

Zur Lösung einer Gleichung mit *zwei* Unbekannten benötigen Sie *zwei* verschiedene Gleichungen; etwa zu der oben stehenden noch die folgende: Y − X = 1.

In allgemeiner Form – bereits in „Computer-Schreibweise" aufbereitet, lauten die Gleichungen:

A*X + B*Y = E
C*X + D*Y = F

Schreiben Sie die obigen Gleichungen in dieser Form, so erhalten Sie:

2*X + 3*Y = 13
−1*X + 1*Y = 1

In der Schule haben Sie gelernt, eine der Unbekannten in einer der beiden Gleichungen zu isolieren und in die zweite einzusetzen. Das geht so:

Aus der Gleichung 2*X + 3*Y = 13
ergibt sich X zu: X = (13 − 3*Y)/2

Das setzen Sie in die zweite Gleichung ein:

−1*((13 − 3*Y)/2) + 1*Y = 1

Rechnen Sie das aus, erhalten Sie für Y = 3 sowie nach einer weiteren Rechnung, für X = 2

Wenn Sie das folgende Programm verwenden, müssen Sie bei der Eingabe für A, B, C, D gut aufpassen! Erinnern Sie sich, oben ist:

A = 2
B = 3
C = −1
D = 1

```
 10 PRINT "PROGRAMM ZUR LOESUNG VON GLEICHUNGEN"
 20 PRINT "MIT ZWEI UNBEKANNTEN NACH SCHEMA"
 30 PRINT
 40 PRINT "A*X + B*Y = E "
 50 PRINT "C*X + D*Y = F "
 60 PRINT "GEBEN SIE NACHEINANDER EIN:"
 70 PRINT
 80 INPUT "A ="; A
 90 INPUT "B ="; B
100 INPUT "C ="; C
110 INPUT "D ="; D
120 INPUT "E ="; E
130 INPUT "F ="; F
140 M = (A*D) - (B*C)
150 IF M = 0 THEN 220
160 X = ((E*D) - (B*F)) /M
170 Y = ((A*F) - (E*C)) /M
180 CLS
190 PRINT "DIE LOESUNG LAUTET:"
200 PRINT "X = "; X , , , "Y = " ; Y
210 PRINT "=====================================
======"
215 END
220 PRINT "FUER DIE EINGEGEBENEN GROESSEN GIBT ES KEINE LOESUNG"
230 END
```

14.2.2 Sie lösen Gleichungen mit drei Unbekannten

Hierfür lautet das allgemeine Schema:

A*X + B*Y + C*Z = D

Für die Größen X = 1, Y = 2, Z = 3 erhalten sie unten drei Gleichungen, die Sie zur Probe durch Einsetzen der Werte für X, Y, Z fast im Kopf rechnen können:

1. Gleichung : 2*X + 1*Y + 3*Z = 13
2. Gleichung : 1*X + 1*Y + 1*Z = 6
3. Gleichung : 1*X + 1*Y − 1*Z = Ø

Das folgende Programm fragt zunächst für jede der drei Gleichungen nach dem obigen Schema nach den Größen für A, B, C und D.

Für unser Beispiel lauten diese:

	A =	B =	C =	D =
1. Gleichung	2	1	3	13
2. Gleichung	1	1	1	6
3. Gleichung	1	1	−1	Ø

```
 5Ø PRINT "GEBEN SIE DIE ERSTE GLEICHUNG EIN! (A, B, C, D)"
 6Ø INPUT A1, B1, C1, D1
 7Ø PRINT "GEBEN SIE DIE ZWEITE GLEICHUNG EIN! (A, B, C, D)!"
 8Ø INPUT A2, B2, C2, D2
 9Ø PRINT "GEBEN SIE DIE DRITTE GLEICHUNG EIN! (A, B, C, D)!"
1ØØ INPUT A3, B3, C3, D3
11Ø E 1 = ((B1*A2) /A1) − B2
12Ø E 2 = ((C1*A2) /A1) − C2
13Ø E 3 = ((B1*A3) /A1) − B3
14Ø E 4 = ((C1*A3) /A1) − C3
15Ø E 5 = ((E1*E4) − (E2*E3))
16Ø IF E5 = Ø THEN 25Ø
17Ø E 6 = ((D1*A2) /A1) − D2
18Ø E 7 = ((D1*A3) /A1) − D3
19Ø Y = ((E6*E4) − (E2*E7)) / E5
2ØØ Z = ((E1*E7) − (E6*E3)) / E5
21Ø X = (D1/A1) − ((B1/A1) * Y) − (C1/A1) * Z
22Ø PRINT "DIE LOESUNG LAUTET:"
23Ø PRINT "X ="; X
24Ø PRINT "Y ="; Y
245 PRINT "Z ="; Z
248 END
25Ø PRINT "FALSCHE GROESSEN EINGEGEBEN!"
26Ø END
```

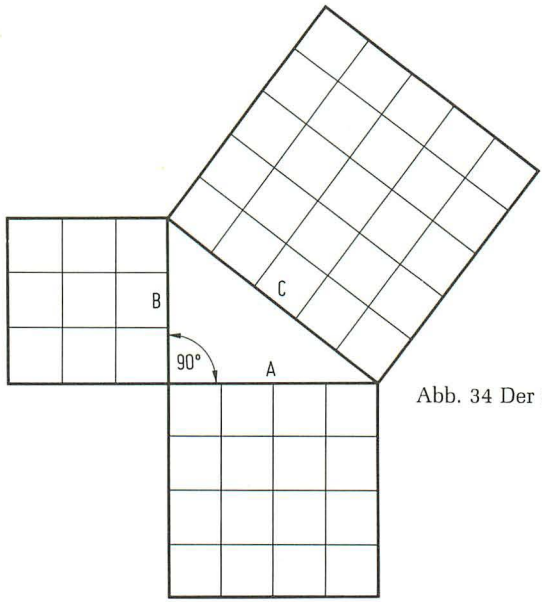


Abb. 34 Der Satz des Pythagoras

$A^2 + B^2 = C^2$

14.2.3 Was ist ein rechtwinkliges Dreieck?

Pythagoras war ein griechischer Philosoph, der um 580 v. Chr. auf Samos geboren wurde. Einen seiner bekanntesten Lehrsätze kennt inzwischen fast jedes Schulkind: den „Satz des Pythagoras". Er lautet: „In einem rechtwinkligen Dreieck ist die Summe der Quadrate über den Katheten gleich dem Quadrat über der Hypothenuse."

Die *Abb. 34* macht diesen Zusammenhang deutlich; wenn Sie Lust haben, können Sie die Richtigkeit durch Auszählen der Quadrate in der Abbildung „beweisen".

Natürlich kann man das auch umkehren und – zu Recht – behaupten: ..."..Ist $A^2 + B^2$ gleich C^2, dann ist der von den Seiten A und B eingeschlossene Winkel ein „rechter Winkel"; d. h., er mißt 90°.

Das folgende Programm soll diesen Nachweis führen:

```
 5 CLS
1Ø PRINT"GEBEN SIE NACHEINANDER DIE DREI SEITEN EIN!"
15 PRINT"DIE GROESSTE SEITE ZULETZT!"
2Ø INPUT A,B,C
25 D = A ↑ 2 + B ↑ 2 : E = C ↑ 2
4Ø IF D = E THEN PRINT"DER WINKEL IST 9Ø GRAD" ELSE PRINT"DER
   WINKEL IST NICHT 9Ø GRAD"
```

Falls Ihr Computer nicht über IF...THEN...ELSE verfügt, müssen Sie schreiben:

```
40 IF D = E THEN 60
50 PRINT"DER WINKEL IST NICHT 90 GRAD":END
60 PRINT"DER WINKEL IST 90 GRAD":END
```

Lassen Sie das ein paarmal laufen, wobei Sie u. a. folgende Werte eingeben:

A =	B =	C =
3	4	5
30	40	50

Bevor Sie an mir, an sich oder an dem Programm verzweifeln, werde ich Ihnen verraten, warum ich Ihnen überhaupt ein so anspruchsloses Programm vorsetze:
Mein Computer hat einmal gerechnet:

A = 30	B = 40	C = 50
A^2 = 900.001	B^2 = 1600	C^2 = 2500

Und nun erkennen Sie mit bloßem Auge, warum das Ergebnis Sie nicht befriedigt: Mit den obigen Werten ist $A^2 + B^2$ eben NICHT GLEICH C^2!

Der Grund für dieses falsche Ergebnis liegt in der Rechengenauigkeit (oder Ungenauigkeit) des Computers! In dem obigen Programm beheben Sie diese Schwäche mit:

```
30 D = INT(D) : E = INT(E)
```

Hatte ich Ihnen nicht schon früher – wenn auch aus anderem Grund – geraten, möglichst mit INT(...) zu arbeiten? Hier haben Sie noch einen Grund!

Wenn Sie sich schon mit dem alten Herrn Pythagoras beschäftigen: Vielleicht gibt es eine Nutzanwendung? Stellen Sie sich vor, Sie haben nur die Länge einer Kathete und die der Hypothenuse. Wie lang muß dann die zweite Kathete sein, damit der von beiden Katheten eingeschlossene Winkel 90° wird? Das führt zu folgendem Mini-Programm:

```
10 INPUT"KATHETE =";B
20 INPUT"HYPOTHENUSE =";C
30 D = INT(B ↑ 2) : E = INT(C ↑ 2) : F = E - D
40 PRINT"IM RECHTWINKLIGEN DREIECK IST DIE 2. KATHETE :";SQR(F)
```

14.2.4 Sie untersuchen die Gleichung: Y = A * X + B

Diese Gleichung beschreibt den Verlauf einer Geraden. Bevor ich das wortreich und umständlich zu erklären versuche, machen Sie mit dem folgenden Programm ein paar Übungen. Spielen Sie dieses Programm sechsmal durch; und zwar mit den Werten für X1, X2, A und B, die in der *Tabelle 7* aufgeführt sind.

Tabelle 7: Eingabegrößen für das folgende Programm

	Eingaben für:					
Beispiel	X 1	X 2	A	B	Formel:	Ergebnis
1	−5	5	1	Ø	Y = 1*X	①
2	−5	5	2	Ø	Y = 2*X	②
3	−5	5	Ø	5	Y = B	③
4	−5	5	2	2	Y = 2*X+2	④
5	−5	5	−2	−2	Y = −2*X−2	⑤
6	−5	5	−2	Ø	Y = −2*X	⑥

```
 1Ø CLS
 15 PRINT "UEBUNG ZUR BEZIEHUNG : Y = A * X + B"
 2Ø PRINT "=====================================
  =====
 3Ø INPUT "KLEINSTER WERT FUER X = "; X 1
 4Ø INPUT "GROESSTER WERT FUER X = "; X 2
 5Ø INPUT "FAKTOR A FUER A * X = "; A
 6Ø INPUT "WERT FUER B = "; B
 7Ø CLS
 8Ø PRINT "X=", "A=", "B=", "Y= "
 9Ø PRINT "=====================================
  =====
1ØØ FOR X = X 1 TO X 2
11Ø Y = A * X + B
12Ø PRINT X, A, B, Y
13Ø NEXT X
```

Die Ergebnisse des obigen Programms sind in der folgenden *Abb. 35* grafisch dargestellt.

Fassen wir daraus die hier interessierenden Ergebnisse zusammen:

- Ist A = Ø, dann ist Y = B; die Gerade verläuft *horizontal. (Gerade 3)*
- *Ist A positiv,* dann verlaufen die Geraden von links-unten nach rechts-oben. (Geraden 1 und 2)

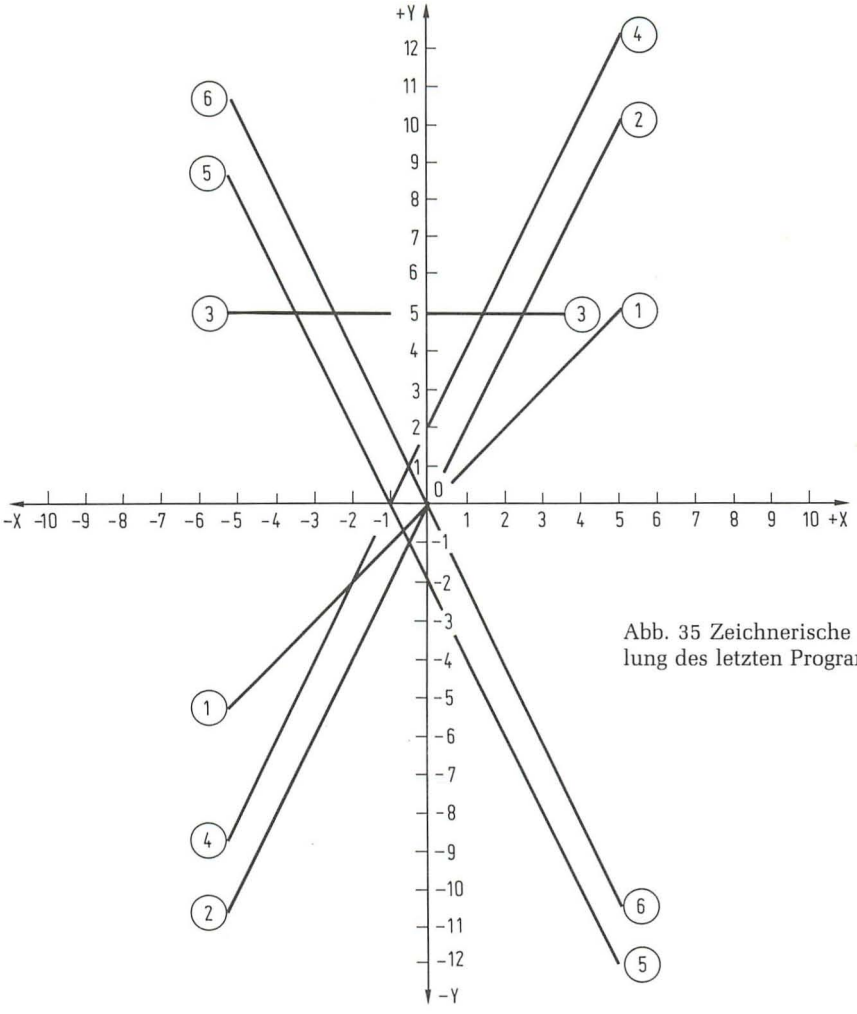


Abb. 35 Zeichnerische Darstellung des letzten Programms

– Ist A *negativ*, dann verlaufen die Geraden von links-oben nach rechts-unten. (Geraden 5 und 6)
– Ist B = ∅, dann verlaufen die Geraden durch den Nullpunkt. (Geraden 1, 2 und 6)

Vereinfachen wir die Auswertung der Geraden aus Abb. 32 noch weiter, dann entdecken wir drei „Typen" von Geraden:
– Horizontal verlaufende
– Mit der Größe X fallende
– Mit der Größe X steigende

Diese drei Typen von Geraden aus der Beziehung Y = A * X + B sind in *Abb. 36* noch einmal schematisch dargestellt.

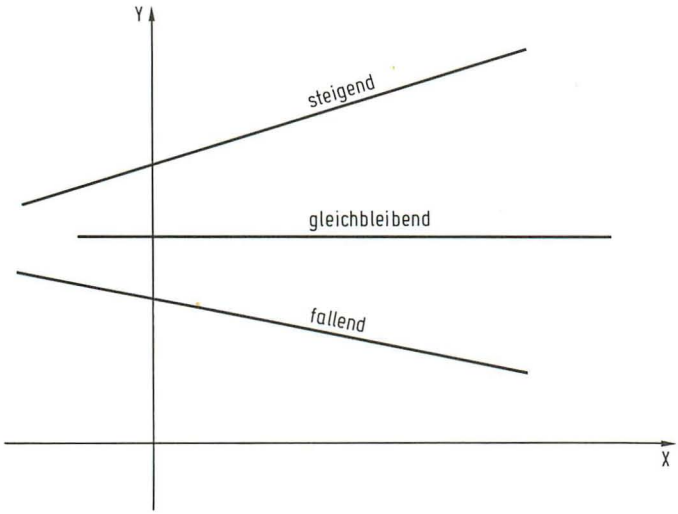


Abb. 36 Tendenzen der Geraden Y = A * X + B

Nun können Sie sich sicherlich vorstellen, daß in einer (Meß-)Aufgabe, bei der etwa in regelmäßigen Zeitabschnitten (das entspricht in den Beispielen X) die Temperatur (das entspricht in den Beispielen Y) gemessen wurde, lediglich interessiert, ob die Temperatur über der Zeit gleichbleibt, ansteigt oder abfällt. Sie können so etwas den „Kurven" in Abb. 36 ansehen. Ihr Computer kann das nicht. Er muß das aus den eingegebenen Werten errechnen. Wir er das machen könnte, läßt sich aus dem Verlauf Ihrer bisherigen Übungen zu diesem Thema leicht ableiten:

Er braucht sich nur die „Größe A" – die die „Steigung" der Geraden bestimmt – „anzusehen". Kommt er dabei zu dem Schluß:

- Ist A = Ø, dann „weiß" er, daß die Gerdade horizontal verläuft; die Tendenz also „gleichbleibend" ist.
- Ist A *positiv*, dann ist die Tendenz steigend...
- Ist A *negativ*, dann ist sie fallend.

In dem folgenden Programm versucht der Computer genau das herauszufinden. In den Zeilen 4Ø, 5Ø ist die Formel Y = A*X + B „nach A umgestellt"; in den Zeilen 6Ø und 7Ø wird das Vorzeichen von A durch die SGN(..)-Funktion geprüft.

14.2.5 Die Nutzanwendung: Sie ermitteln die Tendenz von Meßwerten.

```
 5 CLS
1Ø X1 = 2 : X2 = 6 : B = 5
2Ø INPUT "ERSTEN WERT FUER Y EINGEBEN!"; Y1
25 INPUT "ZWEITEN WERT FUER Y EINGEBEN!"; Y2
```

Tabelle 8

Beispiel	Gleichung	EINGABEN Y1	Y2	ERGEBNIS
1	Y = .5*X+5	6	8	TENDENZ STEIGEND
2	Y = B	5	5	TENDENZ GLEICHBLEIBEND
3	Y = –.5*X+5	4	2	TENDENZ FALLEND

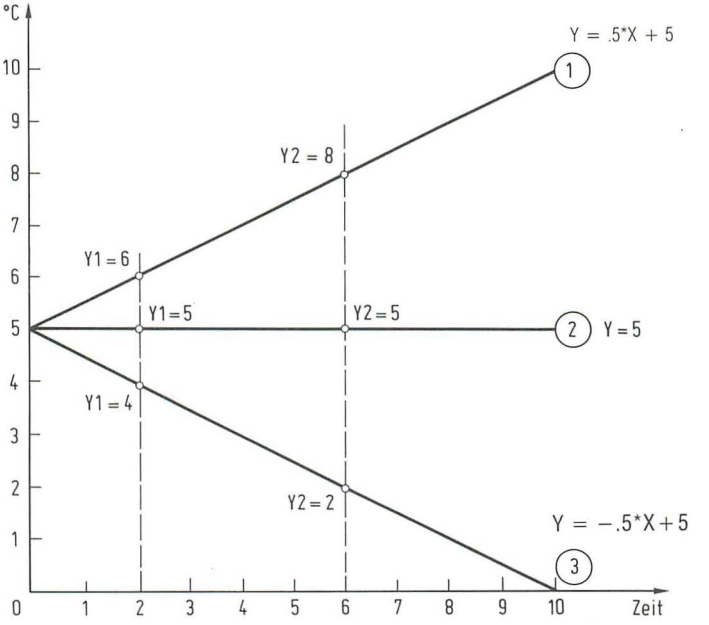


Abb. 37
Geraden aus der Übung; wie ist die Tendenz?

```
40 A1 = (Y1–B)/X1
50 A2 = (Y2–B)/X2
60 IF SGN(A1) = 1 AND SGN(A2) = 1 THEN 100
70 IF SGN(A1) = –1 AND SGN(A2) = –1 THEN 110
80 PRINT "TENDENZ GLEICHBLEIBEND": END
100 PRINT "TENDENZ STEIGEND": END
110 PRINT "TENDENZ FALLEND": END
```

Geben Sie zur Übung die Werte ein, die in der *Tabelle 8* stehen! Hierzu auch *Abb. 37*.

Wenn Sie das letzte Programm mit anderen, beliebigen Werten für Y1 und Y2 weiterspielen, wird es nicht mehr in jedem Falle zu genauen Aussagen kommen. Anders herum: Es erkennt die Tendenzen der Geraden NUR DANN exakt, wenn die Y-Werte wie die Perlen hübsch auf einer Geraden liegen. So etwas können Sie

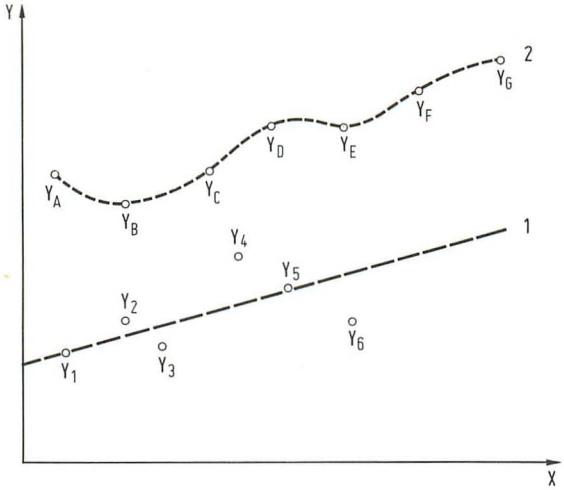


Abb. 38
Die Meßpunkte liegen nicht mehr auf einer Geraden

aber in der Regel kaum erwarten; denken Sie nur an das Beispiel „Temperaturverlauf über der Zeit".

Im Gegenteil: Sie müssen damit rechnen, daß die einzelnen Meßwerte sich so verteilen, wie das die Y1...Y6 bzw. die YA...YG der *Abb. 38* tun.

Wiederum erkennen Sie mit bloßem Auge, daß die Tendenz beider „Punktfolgen" offenbar steigend ist. Richtig vermutet: der Computer erkennt das nicht. Der Computer muß das errechnen. Sie sehen in Abb. 38 durch die Werte Y1...Y6 eine Gerade gestrichelt; so etwa hatte ich den Verlauf der Geraden „interpoliert"; „gemittelt".

Der Computer muß nun gewissermaßen die Gerade aus den Einzelpunkten erst annähernd ermitteln, um daraus das Vorzeichen von A zu erfahren. Wie der Computer „das macht", möchte ich Ihnen nicht im Detail zumuten; ich habe im folgenden A „abgeleitet", dabei aber statt der exakten mathematischen Formel *in Worten* beschrieben, mit welchen Operationen der Computer zu der hier einzig interessierenden Größe A kommt. Wollen Sie's mir abnehmen?

N ist die *Anzahl* der zusammengehörenden Werte*paare* X, Y. (im Beispiel ist N = 1Ø!)
Ferner sind die X gleiche (Zeit-)Intervalle 1, 2, 3... sowie Y die (Meß-)Werte.

A ist der zu X gehörende Faktor, dessen *Vorzeichen* Auskunft über die *Tendenz* der zu *„konstruierenden"* Funktion Y = A * X + B gibt.

$$A = \frac{(1/N-1)^* \text{ (Summe aller } X^*Y - 1/N \text{ * (Summe aller X * Summe aller Y)}}{(1/N-1)^* \text{ (Summe aller } X_{\text{Quadrat}} - 1/N \text{ * ((Summe aller X) zum Quadrat))}}$$

für die weitere Behandlung gilt:
D = Summe aller X * Y
X1 = Summe aller X
Y1 = Summe aller Y
E = Summe aller $X_{Quadrat}$
(X1)↑2 = (Summe aller X)^2

damit vereinfacht sich die Formel zu

$$A = \frac{1/N-1 * (D - 1/N * (X\ 1 * Y\ 1))}{1/N-1 * (E - 1/N * (X\ 1|2))}$$

wird in diesem Ausdruck der Zähler mit U,
der Nenner mit V bezeichnet, dann wird

$$A = \frac{U}{V}$$

Dieses Kauderwelsch wird in folgendes Programm umgesetzt, das natürlich „computergerecht" sein muß!

```
  5 CLS
 10 N = 10
 15 D = 0 : X 1 = 0 : Y 1 = 0 : E = 0
 20 PRINT "X-WERT =", "Y-WERT ="
 30 PRINT "=================================
  ====="
 40 FOR Z = 1 TO N
 50 PRINT X,
 60 INPUT "WERT FUER Y = "; Y
 80 D = D + X * Y
 90 X 1 = X 1 + X
100 Y 1 = Y 1 + Y
110 E = E + X ↑ 2
115 NEXT Z
120 U = (1/N-1) * (D - 1/N * (X 1 * Y 1))
130 V = (1/N-1) * (E - 1/N * (X 1 ↑ 2))
140 A = U/V
150 PRINT "ZUR KONTROLLE : A = "; A
155 IF SGN (A) = 0 THEN 180
160 IF SGN (A) = 1 THEN 190
170 PRINT "DIE TENDENZ IST FALLEND" : END
180 PRINT "DIE TENDENZ IST GLEICHBLEIBEND" : END
190 PRINT "DIE TENDENZ IST STEIGEND" : END
```

Und wieder habe ich Ihnen zur Übung in der *Tabelle 9* Werte vorgeschlagen, die Sie in das vorige Programm einsetzen sollen.

Tabelle 9: Eingabewerte für das „Tendenzprogramm“

	EINGABEN FÜR Y		
	Beispiel 1	Beispiel 2	Beispiel 3
X =	Y =	Y =	Y =
1	−4	4	3
2	−2	4	2
3	−3	4	.5
4	−3	4	−.3
5	Ø	4	−1
6	Ø	4	−1.8
7	1	4	−2.3
8	3	4	−2.8
9	2.5	4	−3.1
1Ø	5	4	−3.5

Wenn Sie mit dem Gelernten „freihändig“ umgehen wollen, dann behalten Sie im Gedächtnis:

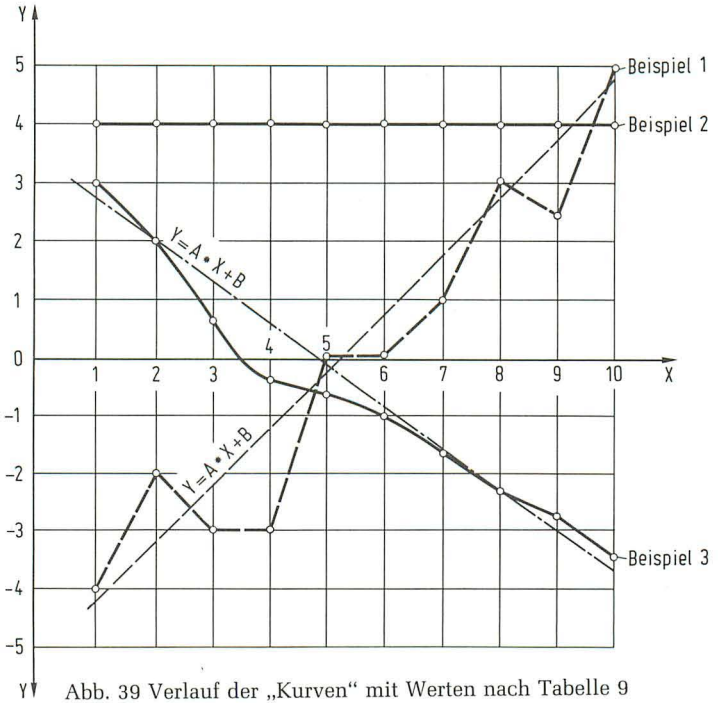


Abb. 39 Verlauf der „Kurven“ mit Werten nach Tabelle 9

- Sie erfassen *Meßwerte Y* (z. B. die Raumtemperatur) in gleichen Abständen X (z. B. jede Minute ein Wert)
- Dann müßten Sie also für X fortlaufend 1, 2, 3, 4, 5... eingeben, für Y die Temperatur, die Sie messen.
  Gehen Sie mit den Werten für Y nicht zu willkürlich um; denken Sie daran, daß unsere ganzen Mühen von dem Zusammenhang Y = A * X + B abgeleitet sind, die Punkte Y also (annähernd) auf einer *Geraden* liegen müssen. (Beispielsweise können bei gleichen X die Y ja auch auf einer Sinuskurve liegen! Aber das ist NICHT unser Thema!)

Zum Schluß zeige ich Ihnen in *Abb. 39* noch den Verlauf der drei Kurven, die Sie mit den Werten der Tabelle 9 erhalten.

14.3 Programme aus der Betriebswirtschaft

14.3.1 Kapitalverzehr

Stellen Sie sich vor, Sie erben 100 000.– DM und legen dieses Kapital zu einem Zinssatz von 10 % bei Ihrer Bank an. Am Ende des ersten Anlagejahres schreibt Ihnen die Bank 10 000.– DM Zinsen gut. Sie können – aus was für Gründen auch immer – von Ihren Zinsen nicht leben; Sie heben den Betrag von 20 000.– DM für das nächste Jahr von Ihrer Bank ab. Es verbleiben als Kapital nur noch 90 000.– DM. Wenn Sie diesen Raubbau an Ihrem Kapital weiterbetreiben, ist Ihr stolzes Vermögen bereits am Ende des 7. Jahres auf magere 5128,29 DM zusammengeschmolzen – wie Ihnen das folgende Programm gleich vorrechnen wird.

```
5 CLS
10 PRINT"KAPITAL-VERZEHR"
20 PRINT"*********************************************************
********"
30 INPUT"VORHANDENES KAPITAL";K
40 INPUT"JAEHRLICHE ZINSEN IN %";P
50 PRINT"ENTNAHME-ART : ........................... JAEHRLICH = 1
.................................................... MONATLICH = 2"
60 INPUT Q
70 ON Q GOTO 100,200
100 INPUT"WELCHEN BETRAG WOLLEN SIE JAEHRLICH ENTNEHMEN";B
105 CLS
110 PRINT"JAHR",,,"REST-KAPITAL"
120 PRINT"=========================================
======"
```

```
125 N = 1
130 K = K*(1+P/100) : K = K – B : K = INT(K*100 +.5)/100
140 PRINT N,,,K
150 K = K*(1+P/100) : K = K – B : K = INT(K*100 +.5)/100
160 N = N +1
170 IF K < = 0 THEN 190
180 GOTO 140
190 END
200 INPUT"WELCHEN BETRAG WOLLEN SIE MONATLICH ENTNEHMEN";B
205 CLS
210 PRINT"MONAT",,,"REST-KAPITAL"
220 PRINT"=====================================
======"
230 K = K * (1 + P/1200) : K = K – B : K = INT(K * 100 + .5)/100
240 PRINT N,,,K
250 N = N + 1
260 K = K * (1 + P/1200) : K = K – B : K = INT(K* 100 + .5)/100
270 IF K < = 0 THEN 290
280 GOTO 240
290 END
```

Ein paar Anmerkungen sollen noch zu diesem Programm gemacht werden:

- Bei der geschilderten jährlichen Entnahme wird Ihnen Ihre Bank mit Sicherheit nach ein paar Jahren nicht mehr den gleichen Zins bezahlen, wie für die vollen 100 000.– DM!
- Wenn Sie gar – wie der zweite Teil des Programmes das unterstellt – Ihr Konto monatlich erleichtern, wird der Zins für dieses Geld – das die Bank als Geld mit vierwöchentlicher Kündigung ansieht – nicht berauschend sein. Das ist „Verhandlungssache"!
- Sie erkennen, daß die beiden Programmteile fast identisch sind; die wesentlichen Unterschiede stecken in den Zeilen 130,230 bzw. 150, 260.
- In den vier erwähnten Programmzeilen steht (fast) viermal dasselbe; warten Sie ab, ob sich das nicht im nächsten Programm geschickter programmieren läßt!

14.3.2 Programm zur Ermittlung der Wertminderungs-Rate

Nichts hält bekanntlich ewig; auch Ihr Computer nicht. Rechnen Sie mal zusammen, was Sie investiert haben und lassen Sie sich durch das folgende Programm ausrechnen, wie Ihr Gerät an Wert verliert unter der Annahme unterschiedlicher Lebensdauer – sprich: Nutzungsdauer!

```
5 CLS
10 PRINT"PROGRAMM ZUR ERMITTLUNG VON :"
20 PRINT"WERTMINDERUNGS-RATE : ..................  KENNZIFFER 1"
30 PRINT"WERTMINDERUNGS-BETRAG : ................  KENNZIFFER 2"
50 PRINT:PRINT
60 PRINT"GEBEN SIE ZUNAECHST EIN :"
70 INPUT"ANSCHAFFUNGSWERT IN DM =";A
80 INPUT"NUTZUNGSDAUER IN JAHREN =";N
85 INPUT"RESTWERT IN DM =";E
```

Anmerkung zu Zeile 85: Hier setzen Sie den Betrag ein, den Sie für Ihren Computer noch erhalten, falls Sie ihn zum Ausschlachten an Bastler verkaufen! Weiter im Text:

```
90 PRINT:PRINT
100 PRINT"GEBEN SIE JETZT DIE KENNZIFFER FUER DIE GEWUENSCHTE
          BERECHNUNG EIN !"
110 INPUT Q
120 ON Q GOTO 200,300
200 CLS
250 IF E = 0 THEN E = 1
280 P = 100*(1–((E/A) ↑ (1/N)))
285 P = INT(P*100+.5)/100
288 CLS
290 PRINT"DIE WERTMINDERUNGS-RATE BETRAEGT";
P; "PROZENT PRO JAHR"
295 PRINT"=========================================
======"
296 IF Q = 2 THEN 400
298 END
300 GOTO 250
400 J = 1 : A = A
420 Z = (A*P)/100: Z = INT(Z*100+.5)/100
425 PRINT"DIE WERTMINDERUNGSBETRAEGE IN DEN EINZELNEN JAHREN
BETRAGEN :"
430 PRINT"JAHR",,,"BETRAG"
435 PRINT"=========================================
======"
440 PRINT J,,,Z
450 J = J + 1
455 A = A –(A*P)/100 : A = INT(A*100 +.5)/100
460 Z = (A*P)/100 : Z = INT(Z*100+.5)/100
470 IF J > N THEN 490
480 GOTO 440
490 END
```

Einige Bemerkungen gefällig?

- Auch im Programmteil „Wertminderungs-Betrag" ist die Wertminderungsrate angegeben. Die Zeilen 25Ø...29Ø sind also in beiden Programmblöcken genutzt. Vergleichen Sie auch die Zeilen 296,3ØØ!!
- Falls Sie Ihren Computer „auf Null" abschreiben wollen, macht die Zeile 23Ø sicher, daß dadurch das Programm nicht zu Unsinn führt; es bestimmt nämlich rigoros: Wenn Sie E zu Null eingeben, macht der Computer selbständig daraus den (symbolischen) Wert „1,– DM". Das ist bei dem gewählten Rechenverfahren auch notwendig: bei E = Ø würde in Zeile 28Ø/285 ermittelt: P = 1ØØ %! Prüfen Sie das nach!

14.3.3 Abschreibungsmethoden

Der § 154 AktG schreibt den Unternehmungen vor, bei Gegenständen des Anlagevermögens, deren Nutzung (zeitlich) begrenzt ist, die Anschaffungskosten um planmäßige Abschreibungen zu vermindern. Die voraussichtliche Wertminderung durch Veralterung oder nutzungsbedingten Verschleiß muß in einem besonderen Abschreibungsplan erfaßt werden.

Der respekterheischende Hinweis oben auf das Aktiengesetzt läßt vermuten, daß man dabei nicht nach Lust und Laune vorgehen kann. Was mir zunächst die Gelegenheit gibt, eine Vorsichtsmaßnahme erneut in Erinnerung zu bringen (das ist wirklich keine Marotte von mir):

Achten Sie bei der Verwendung von gekauften Programmen darauf, ob Sie diese auch verwenden können! Was haben Sie beispielsweise von einem frisch aus den USA importierten „Abschreibungsprogramm", wenn es zwar amerikanischen, nicht aber den hierzulande geltenden Rechtsvorschriften entspricht? Händel mit dem Finanzamt!

Um die Höhe der planmäßigen Abschreibungen zu ermitteln – was gleich Ihr Computer erledigen wird – müssen Sie wissen:

- Die Abschreibungssumme – das ist der Anschaffungswert, ggf. vermindert um einen verbleibenden Restwert, den Sie beispielsweise als Schrottpreis für Ihre alte Anlage erlösen.
- Die Nutzungsdauer in Jahren, die Sie nicht willkürlich wählen dürfen.
- Die Abschreibungsmethode – das ist das Rechenverfahren, nach der Sie die jährlichen Abschreibungsbeträge ermitteln.
- Welche Methode und Nutzungsdauer Sie wählen dürfen – das sagt Ihnen zur Not Ihr Steuerberater oder Buchprüfer.

Zunächst unterscheidet man bei den Abschreibungsmethoden nach leistungsabhängiger Abschreibung und zeitabhängiger Abschreibung.

● Leistungsabhängige Abschreibung

Stellen Sie sich vor, Sie besitzen eine Maschine, mit der Sie Gehäuse für Computer herstellen. Der Hersteller der Maschine verrät Ihnen, daß Sie mit der Maschine 1 000 000 Gehäuse herstellen können, bis die Maschine erneuert werden muß. Sie können dann die Maschinenkosten – abzüglich Restwert – durch die Anzahl der hergestellten Teile dividieren und jedes hergestellte Stück mit diesem Betrag „belasten".

Nach den gleichen Gesichtspunkten können Sie etwa einen Lastzug nach der gefahrenen Strecke abschreiben. Das folgende Programm hilft Ihnen dabei:

```
 LIST
 5 PRINT "LEISTUNGSABH.-ABSCHREIBUNG"
15 PRINT "=========================
 ======"
20 INPUT "ANSCHAFFUNGSWERT"; A
25 INPUT "RESTWERT"; B
30 C = A-B
35 PRINT "ANZAHL DER LEISTUNGSEINHEITEN"
40 INPUT D
45 E = C/D
47 E = INT (E*100+.5)/100
50 PRINT "ABSCHREIBUNGSBETRAG PRO LEISTUNGSEINHEIT"
65 PRINT "="; E; "DM"
70 PRINT "RESTWERT"
75 PRINT "="; B; "DM"
```

● Zeitabhängige Abschreibung

Hier wird die Abschreibungssumme auf die Nutzungszeit verteilt. Wie das geschieht, ist eine Frage des Rechenverfahrens. Ich stelle Ihnen die wichtigsten vor:

– Lineare Abschreibung
Hier verteilen Sie den Abschreibungsbetrag linear – „gleichmäßig" – auf die Nutzungsdauer. Und zwar nach folgendem Programm:

```
  5 PRINT "LINEARE ABSCHREIBUNG"
 15 PRINT "=====================
  ====="
 20 PRINT "ABSCHREIBUNGS-ZEITRAUM"
 30 INPUT N
 35 PRINT "ANSCHAFFUNGSWERT:"
 40 INPUT A
 45 PRINT "RESTWERT:"
 50 INPUT E
 55 J = 1 : A = A
 60 P = (A-E) / N
 65 P = INT (P * 100 + .5)/100
 66 X = A-P : X = INT (X*100 + .5) / 100
 67 PRINT "ABSCHREIBUNGSBETRAG"
 68 PRINT "=";P;" DM/JAHR": PRINT
 70 PRINT "JAHR", "RESTWERT"
 75 PRINT"================
  ====="
 80 PRINT J, X
 90 J = J + 1 : A = A-P : P = P : X = A-P : X = INT(X*100+.5)/100
100 IF J>N THEN 120
110 GOTO 80
120 PRINT "RESTWERT"
130 PRINT "="; E; "DM"
140 END
```

Die lineare Abschreibung – obwohl zulässig für alle Wirtschaftsgüter – ist nicht sonderlich gefragt, da man meist in den ersten Jahren höhere Abschreibungsbeträge realisieren möchte. Das führt zunächst zur:

– Arithmetisch-degressive(n) Abschreibung

Hier fällt der Abschreibungsbetrag in jedem Jahr um einen gleichmäßigen Betrag, der sich aus einer „arithmetischen Reihe" ergibt. Sie verstehen das am besten durch folgendes Beispiel:

– Sie wollen einen Betrag von 150 000.– DM in N = 5 Jahren abschreiben.
– Sie bilden die Summe der Abschreibungsjahre: 1 + 2 + 3 + 4 + 5 = 15
– Sie dürfen dann abschreiben:

im ersten Jahr	5/15	=	50 000.– DM
im zweiten Jahr	4/15	=	40 000.– DM
im dritten Jahr	3/15	=	30 000.– DM
im vierten Jahr	2/15	=	20 000.– DM
im fünften Jahr	1/15	=	10 000.– DM

Das ergibt nach Adam Riese 150 000.– DM in fünf Jahren.
Diese Aufgabe erledigt das folgende Programm. Besonders hinweisen möchte ich auf die Programmzeile 9Ø, in der trickreich die „Jahressumme X" ermittelt wird. Vollziehen Sie das nach für 5 Jahre!

X = (N * (N + 1))/2
X = (5 *(5 + 1))/2
X = 15

Und nun für 10 Jahre!

X = (10 * (10 + 1))/2
X = 55

● **Arithmetisch-degressive Abschreibung**

```
 LIST
 1Ø PRINT "ABSCHREIBUNG NACH DER ARITHEM.-DEGRESSION"
 4Ø PRINT "=================================
  ====="
 5Ø PRINT "ANSCHAFFUNGSWERT"
 6Ø INPUT A
 7Ø PRINT "ABSCHREIBUNGS-ZEITRAUM"
 8Ø INPUT N
 9Ø X = (N * (N+1))/2
1ØØ Z = A/X
11Ø I = 1
12Ø B = Z * N
125 B = INT (B*1Ø)/1ØØ
13Ø A = A-B
14Ø PRINT "JAHR", "ABSCHR.-BETRAG"
16Ø PRINT "=====================
 ======"
17Ø PRINT I, B
18Ø I = I+1:N=N-1:B=Z*N:A=B-A
185 B =INT (B*1ØØ+.5)/1ØØ
19Ø IF N < 1 THEN 21Ø
2ØØ GOTO 17Ø
21Ø END
```

Aus einem mir nicht ersichtlichen Grund wird die arithmetisch-degressive Abschreibung auch unter dem Namen „digitale Abschreibung" gehandelt. Sie ist

leider nur selten anwendbar, da sie steuerlich nur zulässig ist, wenn sich für das erste als auch für das erste bis dritte Jahr keine höheren Beträge ergeben als bei der:

● **Geometrisch-degressive(n) Abschreibung**

Diese Rechenmethode tauchte schon bei dem Programm „Wertminderung" auf; die Rechenverfahren und Formeln sind identisch. Beachten Sie deshalb auch Zeile 4Ø!

```
  2 PRINT "DEGRESSIVE ABSCHREIBUNG"
  6 PRINT "=======================
  ====="
 1Ø PRINT "ABSCHREIBUNGSZEIT"
 15 INPUT N
 2Ø PRINT "ANSCHAFFUNGSWERT"
 25 INPUT A
 3Ø PRINT "RESTWERT"
 35 INPUT E
 4Ø IF E=Ø THEN 6Ø
 5Ø GOTO 7Ø
 6Ø E=1
 7Ø J=1:A=A
 75 P=1ØØ*(1-((E/A) ↑ (1/N)))
 76 P=INT(P*1ØØ+.5)/1ØØ
 8Ø Z=(A*P)/1ØØ:Z=INT (Z*1ØØ+.5)/1ØØ
 85 X=A-Z:X=INT(X*1ØØ+.5)/1ØØ
 9Ø PRINT "ABSCHR.-FAKTOR"
 95 PRINT"=";P
1ØØ PRINT"=========================
======"
11Ø PRINT "JAHR", "ABSCHR.-BETRAG"
12Ø PRINT"=========================
======"
13Ø PRINT J, Z
135 J=J+1
14Ø A=A-(A*P)/1ØØ:A=INT(A*1ØØ+.5)/1ØØ
15Ø Z=(A*P)/1ØØ:Z=INT(Z*1ØØ+.5)/1ØØ
16Ø X=A-Z:X=INT(X*1ØØ+.5)/1ØØ
17Ø IF J > N THEN 19Ø
18Ø GOTO 13Ø
19Ø PRINT "RESTWERT"
2ØØ PRINT "=";E;" DM"
```

Nun achtet das Finanzamt peinlich genau darauf, daß Sie nicht zu wenig Steuern bezahlen – sprich: zu schnell oder zu hoch abschreiben. Das letzte Programm ist deshalb in seiner „chemischen Reinheit" nicht brauchbar, weil steuerlich hinsichtlich Nutzungsdauer und Abschreibungssatz Höchstwerte nicht überschritten werden dürfen.

Damit Sie dennoch bei Bedarf „auf Null" abschreiben können, ist in einem bestimmten Übergangsjahr von geometrisch-degressiver auf lineare Abschreibung „umzuschalten", wie es die *Tabelle 10* zeigt. Diese Tabelle ist in das folgende Programm „eingearbeitet". Studieren Sie (für Ihr weiteres Fortkommen mit BASIC), wie jeweils die richtigen Daten zueinanderfinden!

```
10 PRINT "DEGRESSIVE ABSCHREIBUNG"
30 PRINT "*****************************************************
********"
40 DIM J(16):DIM P(16):DIM L(16)
50 FOR N=1 TO 16
60 READ J(N), P(N), L(N)
```

Tabelle 10: Regeln für die Abschreibung

Nutzungsdauer (Jahre)	Abschreibungssatz (%)	Übergangsjahr auf lineare Abschreibung
unter 5	nicht zulässig	–
5–9	25	8.
10	25	8.
11	22.73	9.
12	20.83	9.
13	19.23	10.
14	17.86	11.
15	16.67	11.
16	15.63	12.
17	14.71	12.
18	13.89	13.
19	13.16	14.
20	12.5	
über 20	keine Angaben	

Anmerkung:
Die Daten dieser Tabelle stehen in der DATA-Liste des folgenden Programms wie folgt:
Nutzungsdauer = J (N)
Abschreibungssatz = P (N)
Übergangsjahr = L (N)

```
 7Ø NEXT N
 8Ø PRINT "ANSCHAFFUNGSWERT"
 9Ø INPUT A
12Ø PRINT "AUSWAHLTABELLE:"
13Ø PRINT "FUER ABSCHR.-ZEIT"
14Ø PRINT "==================
======"
15Ø PRINT "JAHRE", "KENNZIFFER"
155 PRINT"==================
======"
16Ø PRINT "5", "1"
162 PRINT "6", "2"
164 PRINT "7", "3"
166 PRINT "8", "4"
168 PRINT "9", "5"
17Ø PRINT "1Ø", "6"
172 PRINT "11", "7"
174 PRINT "12", "8"
176 PRINT "13", "9"
178 PRINT "14", "1Ø"
18Ø PRINT "15", "11"
182 PRINT "16", "12"
184 PRINT "17", "13"
186 PRINT "18", "14"
188 PRINT "19", "15"
19Ø PRINT "2Ø", "16"
2ØØ PRINT "GEWUENSCHTE KENNZAHL EINGEBEN!"
2Ø4 INPUT N
21Ø J=1:A=A
22Ø Z=(A*P(N))/1ØØ
225 Z=INT (Z*1ØØ+.5)/1ØØ
23Ø X=A-Z
235 X=INT (X*1ØØ+.5)/1ØØ
24Ø PRINT "DEGRESSIVE ABSCHREIBUNG"
242 IF L(N)=Ø THEN 246
244 PRINT "BIS ZUM"; L(N)-1; "-TEN JAHR"
246 PRINT "MIT ABSCHR.-FAKTOR=";P(N)
25Ø PRINT"============================
======"
26Ø PRINT "JAHR", "ABSCHR.-"BETRAG"
27Ø PRINT"========================
======"
```

```
275 PRINT J,Z
280 J=J+1
285 A=A-(A*P(N))/100
286 A=INT(A*100+.5)/100
290 Z=(A*P(N))/100
295 Z=INT(Z*100+.5)/100
300 IF L(N) =0 THEN 322
310 IF J=L(N)THEN 330
320 GOTO 275
322 X=A-Z
324 D=J (N)
326 IF J > D THEN 480
328 GOTO 275
330 X=A-Z
340 J1=(J(N)-L(N))+1
350 R=X/J1
360 R=INT (R*100+.5)/100
370 PRINT "UEBERGANG AUF"
375 PRINT "LINEARE-ABSCHREIBUNG AB DEM"
385 PRINT L(N); "-TEN JAHR"
390 PRINT "RESTWERT"
395 PRINT "=";X;" DM"
400 PRINT "JAHR", "ABSCHR.-BETRAG"
415 PRINT"=======================
======"
420 PRINT L(N),R
430 L(N)=L(N)+1
440 IF L(N) > J(N) THEN 460
450 GOTO 420
460 PRINT "RESTWERT= 0 DM"
470 END
480 PRINT "RESTWERT=";X;" DM"
490 END
1000 DATA 5, 25, 0
1005 DATA 6, 25, 0
1010 DATA 7, 25, 0
1015 DATA 8, 25, 0
1020 DATA 9, 25, 0
1025 DATA 10, 25, 8
1030 DATA 11, 22.73, 8
1035 DATA 12, 20.83, 9
```

```
1040 DATA 13, 19.23, 9
1045 DATA 14, 17.86, 10
1050 DATA 15, 16.67, 11
1055 DATA 16, 15.63, 11
1060 DATA 17, 14.71, 12
1065 DATA 18, 13.89, 12
1070 DATA 19, 13.16, 13
1075 DATA 20 12.5, 14
1560 END
```

Das Programm ist für den PC 100 geschrieben und berücksichtigt dessen Drucker. Wollen Sie die Ausgabe nur auf dem Bildschirm betrachten, dann müssen Sie die Ausgabe unterbrechen, etwa mit:

```
5 CLS
175 PRINT "WEITER?, DANN 'ENTER' DRUECKEN!": INPUTQ : CLS
```

Den gleichen Text noch einmal als Zeile 360 eingeben!

14.3.4 Hochrechnungen

Ich möchte das Erklären der mathematischen Grundlage von Programmen – in Ihrem Interesse – nicht übertreiben. Deshalb nehmen Sie bitte das gleich folgende Programm kommentarlos hin! Schließlich brauchen Sie auch nicht zu wissen, wie Ihr Computer im Detail funktioniert, um ihm per BASIC eine Menge zu entlocken.

Ich möchte Ihnen vielmehr erklären, wie man ein derartiges „Wahrsage"-Programm – das Hochrechnungen ausführt und damit zu Prognosen fähig scheint – sinnvoll nutzt und was man von ihm – dem Programm – besser nicht erwarten sollte.

Wenn Sie das Programm starten, wird es Sie zunächst danach fragen, für wieviele „Perioden" Werte vorliegen. Das bedeutet:

- Perioden deshalb, weil es bei der Art des Programms gleichgültig ist, ob die zeitlichen Abstände der Werte Tage, Monate, Quartale oder Jahre sind – untereinander gleich müssen die Perioden sein!
- Die Werte müssen gewissermaßen gesicherte Fakten sein, Werte die Sie ganz genau und ohne jeden Zweifel kennen. Sind Sie etwa Kaufmann, dann kennen Sie sicher die Umsätze Ihres Hauses für jeden Monat des letzten Jahres. (Wenn nicht, versuchen Sie am besten, schnellstmöglichst ins Pensionärsdasein wegzutauchen, bevor Sie der Konkurs ereilt!)
  Übersetzen Sie einfach „Werte" mit: „Zahlen, die stimmen"!

Diese beiden Anmerkungen sind für die Nutzung des Programms deshalb wichtig, weil es sich anmaßt, auf der Grundlage Ihrer Eingabe eine Vorhersage zu machen. Um beim Kaufmannsbeispiel zu bleiben: Es prognostiziert Ihren Umsatz für

beliebig viele Perioden in der Zukunft und fragt Sie dazu, bis zu welcher Periode hochgerechnet werden soll. Vorher erfahren Sie noch, wie die mittlere Wachstumsrate – vorausgesetzt, es gibt nach Ihren Eingaben eine – über die eingegebenen Perioden war.

Damit wir dem Programm gemeinsam auf die Finger sehen können, mache ich Ihnen zunächst Vorschläge für Ihre Eingabe.

– Beantworten Sie die Frage:
  FUER WIEVIELE PERIODEN LIEGEN WERTE VOR ?
  mit 1Ø
– Dann geben Sie ein:

Periode	Wert
1	5Ø
2	55
3	57
4	6Ø
5	56
6	58
7	62
8	6Ø
9	6Ø
1Ø	63

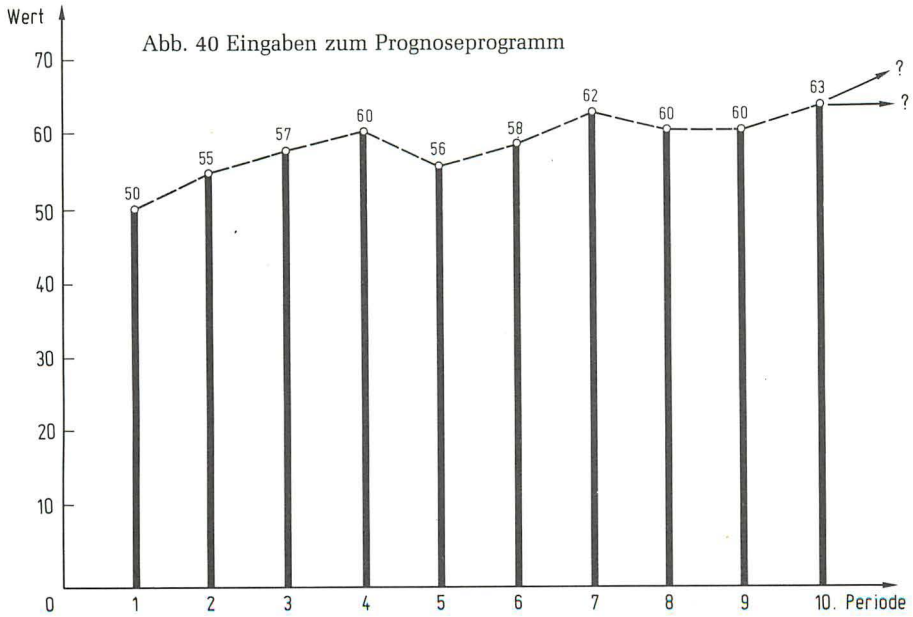


Abb. 40 Eingaben zum Prognoseprogramm

In der folgenden *Abb. 40* zeige ich Ihnen Ihre Eingabe in Form eines Balken-Diagramms; das „liest" sich doch wie die Umsatzkurve eines ganz normalen Unternehmens! Ein wenig konjunkturelles Auf und Ab; aber immerhin über zehn Perioden eine Steigerung von 50 auf 63; um 26 %.

Unmittelbar nach Ihrer letzten Eingabe hat Ihnen der Computer bereits geantwortet:

DIE MITTLERE WACHSTUMSRATE IST 1.88 %

Nun beantworten Sie die Frage:

BIS ZU WELCHER PERIODE SOLL HOCHGERECHNET WERDEN ?

mit 2Ø. Das heißt, Sie wollen über die 10. Periode hinaus – die Sie ja eingegeben hatten – die 11., 12. usw. bis 20. erfahren.

Sie erhalten zur Antwort:

PERIODE	WERT
11	64.25
12	65.46
13	66.69
14	67.95
15	69.23
16	7Ø.53
17	71.86
18	73.23
19	74.6
2Ø	76

Während es im wirklichen Leben wohl nicht ohne Auf und Ab geht, scheint es für Ihren Hellseher – wie Sie leicht überprüfen können – unaufhaltsam aufwärts zu gehen. Das sollte Sie stutzig machen! Wenn Sie einmal zwei beliebige Werte nachrechnen – etwa die von Periode 14 und 15 – dann erkennen Sie, daß zwischen allen ein Abstand ist von ca. 1.8 % – und das hatte Ihnen der Computer als mittlere Wachstumsrate oben aus Ihren Werten der Perioden 1...10 ausgerechnet. In Worte gefaßt, bedeutet das: Der Computer unterstellt, daß in der Zukunft Ihr Geschäft den gleichen Trend nimmt wie in der Vergangenheit. Das sollte Sie wieder stutzig machen, weil es zu schön ist, um wahr zu sein!

Jetzt versuchen Sie einmal, herauszufinden, wie weit man dem überhaupt trauen kann!

Obwohl – wie wir vereinbart hatten – Ihnen alle Werte der ersten zehn Perioden bekannt sind, tun Sie Ihrem Computer gegenüber so, als würden Sie nur die ersten fünf kennen und fragen ihn nach den Werten der Perioden 6...10!

Und das kommt dabei heraus:

Periode	Hochrechnung der Computers	tatsächliche Werte
6	6Ø.98	58
7	62.92	62
8	64.93	6Ø
9	67	6Ø
1Ø	69.13	63

Je weiter der Computer in die Zukunft schaut, umso größer wird offenbar seine „Mißweisung"!

Versuchen Sie, einmal selbständig einen Test zu machen! Geben Sie für zehn Perioden Werte ein, die gleiche Abstände voneinander haben wie 2-4-6-8 usw.

Geben Sie dann für zehn Perioden Werte ein, die fallen wie 8-7-6-5-4... Sie werden erleben:

* Je mehr Perioden bekannt sind...
* Je kürzer man den Vorhersagezeitraum macht...

Um so passabler wird das Ergebnis.
Bei fallenden Werten kommt der Computer zwar zu einem negativen Prozentsatz (Wie die Politiker: „Minus"-Wachstum!), aber negative Geschäftsergebnisse scheint er nicht zu kennen. Er erreicht bei der Vorhersage irgendwann Null und bleibt stur dabei.

Ach ja, eine wichtige Information für Sie: Die Zahlen für die zehn Perioden, die einzugeben ich Ihnen empfohlen hatten, sind exakt die Zahlen des Personalbestandes eines mir bekannten, kleinen Unternehmens für die Perioden 1970...1979.

Rechnet man mit diesen Werten den Personalbestand hoch, dann müßte die Firma im Jahre 1984 über knapp 68 Mitarbeiter verfügen.

In Wirklichkeit hat die Firma im Sommer 1982 Konkurs angemeldet... Ich hoffe, das „sitzt"!

Seien Sie vorsichtig mit Prognosen, die Sie selbst machen. Und vor allem mit solchen, die Sie lesen! Trotz allem: hier folgt das Programm:

```
1Ø CLS
2Ø DIM W(5Ø)
25 PRINT"          PROGNOSE-PROGRAMM"
26 PRINT"=========================================
====="
28 PRINT:PRINT
3Ø PRINT"FUER WIEVIEL PERIODEN LIEGEN WERTE VOR?"
4Ø INPUT N
```

```
50 FOR I = 1 TO N
60 PRINT"PERIODE =";I;
70 INPUT W(I)
80 NEXT I
90 T = LOG(W(1))
100 X = 0
110 FOR I = 2 TO N
120 L = LOG(W(I))
130 T = T + L
140 X = X +(I–1)*L
150 NEXT I
160 A = 6*(2*X/(N–1)–T)/N/(N+1)
170 G = EXP(A)–1 : G=100*(EXP(A)–1) : G = INT(G*100+.5)/100
190 PRINT"DIE MITTLERE WACHSTUMSRATE IST :";G;" %"
210 G=EXP(A)–1
220 W = EXP(T/N–A*(N–1)/2)
230 PRINT"BIS ZU WELCHER PERIODE SOLL HOCHGERECHNET WERDEN ?"
240 INPUT M
250 PRINT"VORAUSSICHTLICHE ENTWICKLUNG DER PERIODENWERTE :"
260 PRINT"PERIODE",,,"WERT"
270 PRINT"==========================================
======"
280 J = N + 1
290 K = N
300 H =(G+1)↑ K
310 W1=W*H : W1=INT(W1*100+.5)/100
320 PRINT J,,,W1
330 J=J+1:K=K+1
335 H=(G+1)↑ K
340 W1=W*H:W1=INT(W1*100+.5)/100
350 IF J > M THEN 370
360 GOTO 320
370 END
```

14.3.5 Währungstabellen und andere

Das folgende Programm rechnet Währungen um; es gibt beispielsweise aus, was 56.– US-Dollar in DM „kosten". Wenn Sie es vor sich sehen, werden Sie es mit Sicherheit überhaupt nicht mehr nutzen können. Weil es für die Wechselkurse am 21. Dezember 1981 gilt, nicht für die heutigen. Sie lernen aber bei der

Programmentwicklung eine neue Technik kennen, die ich Ihnen nicht vorenthalten will.

● **Stringtabelle**

Legen wir zunächst eine Tabelle von Strings an, die später oft genutzt werden.

```
10 A$="          UMRECHNUNGSTABELLE FUER FREMDWAEHRUNGEN"
14 B$="=========================================
====="
20 C$="21. DEZEMBER 1981"
25 D$="DEUTSCHE MARK IN"
30 E$="IN DEUTSCHE MARK"
35 F$="ZUM WEITERLESEN 'E N T E R' DRUECKEN"
40 G$="FUER WEITERE UMRECHNUNGEN TASTE 1 DRUECKEN,DANACH
'E N T E R'"
45 H$="ANGABEN GUELTIG AM :"
50 J$="ERGEBEN :"
55 A1$="DEUTSCHE MARK"
60 A2$="US-DOLLAR"
65 A3$="PFUND STERLING"
70 A4$="IT. LIRE"
75 A5$="SCHWEDENKRONEN"
80 A6$="HOLL. GULDEN"
85 A7$="SPAN. P'TAS"
90 A8$="GRIECH. DRACHMEN"
95 A9$="FRZS. FRANC"
100 B1$="BELG. FRANC"
105 B2$="DAEN. KRONEN"
110 B3$="OEST. SCHILLING"
115 B4$="SCHWEIZ. FRANKEN"
```

● **Bedienungsanleitung**

Die Bedienungsanleitung beschreibt u. a. wie man als Benutzer vorgehen muß, um:

– in das Programm das aktuelle Tagesdatum einzugeben
– bei Kursänderungen neue Umrechnungsfaktoren in das Programm einzugeben.

```
145 CLS
150 PRINT A$ : PRINT B$
155 PRINT"DIESES PROGRAMM FUEHRT WAEHRUNGS-UMRECHNUNGEN DURCH."
160 PRINT"ES GILT FUER DEN AMTLICHEN KURS AM : ";C$
165 PRINT"BEI AENDERUNG VON WECHSELKURSEN GEHEN SIE WIE
    FOLGT VOR :"
```

```
170 PRINT"        * SUCHEN SIE IN DER LINKEN SPALTE DER AUSWAHL-
                    TABELLE DIE KENNZIFFER DER WAEHRUNG, DEREN
                    KURS SICH GEAENDERT HAT !"
175 PRINT"        * ADDIEREN SIE ZU DER KENNZIFFER 1000 !"
177 PRINT"        * BRECHEN SIE DAS PROGRAMM AB DURCH DRUCK AUF
                    DIE TASTE 'B R E A K' !"
178 PRINT:PRINT F$:INPUT Q:CLS
180 PRINT"        * RUFEN SIE MIT DER SUMME (KENNZIFFER + 1000) DIE
                    DATENZEILE AUF ! BEISPIEL: LIST 1017"
185 PRINT"        * ERSETZEN SIE DIE DATEN DURCH DEN HEUTE GUELTI-
                    GEN KURSFAKTOR !"
190 PRINT"        * RUFEN SIE MIT LIST 20 DIE DATUMZEILE AUF UND
                    TRAGEN SIE DAS HEUTIGE DATUM EIN !"
195 PRINT:PRINT"WENN SIE ALLES VERSTANDEN HABEN,DRUECKEN SIE
    'E N T E R'"
200 INPUT Q:CLS
```

Haben Sie das verstanden?

In der gleich folgenden Auswahltabelle ist beispielsweise 1 die Kennziffer für die Umwandlung von DM in US-Dollar. Kennziffer + 1000 ergibt 1001.

Sie rufen also auf:

LIST 1001

Es erscheint:

1001 DATA .4545,2.2

Dieses „Datum" – und alle anderen – ist so zustande gekommen:

Am 21. Dezember 1981 betrug der Kurs DM 2,20 für 1 US-Dollar. Das bedeutet: 1 DM = 1/2.2 US-Dollar. Und 1/2.2=.4545

Hat sich der Kurs heute auf beispielsweise 2.45 geändert, dann ist 1/2.45=.4082

Damit ist die Datenzeile zu ändern in:

1001 DATA .4082,2.45

Rufen Sie LIST 20, so erscheint:

20 C$="21.DEZEMBER 1981"

Haben wir heute den 3. September 1983, so müssen Sie natürlich die ganze Zeile wie folgt überschreiben:

20 C$="3.SEPTEMBER 1983"

● Datenliste

1001 DATA .4545,2.2

1003 DATA .2, 5

Eine Warnung! Damit das Programm funktioniert, müssen Sie ausnahmsweise auch die Zeilen-Nummern genau übertragen! Es würde sonst das oben beschrie-

bene Ändern von Daten zu falschen Ergebnissen führen. Aber auch der im Programm vorgesehen Zugriff auf die Daten käme bei anderer Zeilen-Numerierung durcheinander!

Sie werden gleich klarer sehen! Zunächst einmal weiter:

```
1ØØ5 DATA 444.444,.ØØ225
1ØØ7 DATA 2.2222,.45
1ØØ9 DATA 1.Ø526,.95
1Ø11 DATA 37.Ø37,.Ø27
1Ø13 DATA 21.276,Ø47
1Ø15 DATA 2.3Ø9,.433
1Ø17 DATA 15.625,.Ø64
1Ø19 DATA 3.125,.32
1Ø21 DATA 7.1429,.1399
1Ø23 DATA .9Ø91,1Ø99
```

● Auswahltabelle

Mit dem folgenden Programmteil wählen Sie aus, welche Währung sie umrechnen wollen; etwa DM in it. Lire oder Schweizer Franken in DM.

```
2ØØ CLS
2Ø5 PRINT A$:PRINT B$
```

Es folgen zweizeilige Programm-Blocks:

```
21Ø PRINT @ 128,D$ : PRINT @ 16Ø,A2$
215 PRINT @ 192,A2$:PRINT @ 221 "1":PRINT @ 224,E$:PRINT @ 254,"2"
```

Diese beiden Programmzeilen will ich Ihnen zunächst einmal „übersetzen", d. h., so hinschreiben, wie der Computer sie ausgibt:

DEUTSCHE MARK IN		US-DOLLAR	
US-DOLLAR	1	IN DEUTSCHE MARK	2

Es ist also:

– 1 die Kennziffer für DM in US-Dollar
– 2 die Kennziffer für US-Dollar in DM

Verfügt Ihr Computer nicht über PRINT @, dann setzen Sie die Auswahltabelle nach den Möglichkeiten Ihres Gerätes zusammen!

Der nächste zweizeilige Programmblock lautet:

```
22Ø PRINT @ 256,D$ : PRINT @ 288,A3$
225 PRINT @ 32Ø,A3$:PRINT @ 349,"3":PRINT @ 352,E$:PRINT @ 382,"4"
```

Vergleichen Sie die Zeilen 21Ø/215 mit den Zeilen 22Ø,225! Sie erkennen, welche Strings wiederkehren (D$,E$) und welche neu sind (A3$). Schauen Sie genau hin, damit Sie das Prinzip erkennen; Sie sollen das Programm nämlich jetzt selbständig zu Ende führen! Ich geben Ihnen noch den letzten Zweizeiler wie folgt an:

```
330 PRINT @ 768,D$ : PRINT @ 800,B4$
335 PRINT @ 832,B4$:PRINT $ 861,"23":PRINT @ 864,E$:PRINT @ 894,"24"
```

Damit Sie es nicht ganz so schwer haben, verwenden Sie bei Ihrer Arbeit die folgende Liste:

Pr.-Zeile	PRINT@-Positionen				eingeführte Variable
210	128		160		A2$
215	192	221	224	254	
220	256		288		A3$
225	320	349	352	382	
230	384		416		A4$
235	448	477	480	510	
240	512		544		A5$
245	576	605	608	638	
250	640		672		A6$
255	704	733	736	766	
260	768		800		A7$
265	832	861	864	894	
270	PRINT @ 896,F$				
275	INPUT Q : CLS				
280	128		160		A8$
285	192	221	224	254	
290	256		288		A9$
295	320	349	352	382	
300	384		416		B1$
305	448	477	480	510	
310	512		544		B2$
315	576	605	608	638	
320	640		672		B3$
325	704	733	736	766	
330	768		800		B4$
335	832	861	864	894	
340	PRINT @ 896,F$				
345	INPUT Q : CLS				

● **Programmteil: Lesen der Daten**

```
6 DIM F(24)
7 FOR N = 1 TO 24
8 READ F(N)
9 NEXT N
```

Damit sind die Faktoren für die Umwandlung aus der Datenliste in den Computer eingelesen. Wie Sie leicht überprüfen können, hat z. B.:

F(3) den Wert .2
F(4) den Wert 5 usw.
Beachten Sie, daß zwischen Kennziffern, Faktor F(..) und Datenzeile folgender Zusammenhang besteht: (Beispiel!)
* Die Kennziffern für die Umwandlung Deutsche Mark in Schwedenkronen ist 7
* Der Faktor für die Umwandlung ist F(7)
* Dieser ist in der Datenzeile 1007 abgelegt.
Das ist natürlich mit Absicht so gemacht und erleichtert die Umrechnung bzw. den Programmaufbau sehr.

● Das Umwandlungsprogramm

```
400 PRINT @ 448, "GEBEN SIE DIE KENNZIFFER FUER DIE GEWUENSCHTE
                  UMWANDLUNG EIN !"
405 INPUT N
410 IF N < 1 OR N > 24 THEN 415
412 GOTO 425
415 PRINT"SIE HABEN EINE FALSCHE ZAHL EINGEGEBEN ! 1...24 !"
417 FOR Z = 1 TO 1000 : NEXT
419 CLS : GOTO 400
425 CLS : PRINT @ 512, "GEBEN SIE DEN BETRAG EIN,DER UMGERECHNET
                        WERDEN SOLL !"
430 INPUT A
435 X = A * F(N)
```

Das ganze Umwandlungs-„Programm" besteht also nur aus der Zeile 435!

Mit der von Ihnen eingegebenen Kennziffer für die Umwandlung – die für den Computer N heißt – (Zeile 400/405) gehen Sie jetzt auch in die Ausgabe-Routinen. Und zwar:

```
440 ON N GOTO 500,520,540,580,600,620,640,660,680,700,720,740,760,
780,800,820,840,860,880,900,920,940,960
```

Das ist sicher eine ungewöhnliche Verwendung des ON...GOTO..-Statements!

```
971 INPUT Z
972 IF Z = 1 THEN 205
974 CLS : PRINT @ 400,"ENDE DES UMWANDLUNGS-PROGRAMMS"
976 END
```

● Das Ausgabeprogramm

Das Ausgabeprogramm besteht aus insgesamt 24, zum großen Teil identischen Blocks, von denen die ersten beiden so lauten:

```
500 CLS : PRINT @ 400,A;A1$
505 PRINT @ 464,J$ : PRINT @ 528,X;A2$ : PRINT @ 704,H$;C$
510 PRINT @ 832,G$ : GOTO 971
520 CLS : PRINT @ 400,A;A2$
525 PRINT @ 464,J$ : PRINT 528,X;A1$ : PRINT @ 704,H$;C$
530 PRINT @ 832,G$ : GOTO 971
```

Sie sehen also, daß sich die beiden Blocks nur in den Strings A1$ und A2$ sowie in ihrer Plazierung unterscheiden. Sie sind sicher in der Lage, unter Berücksichtigung der Zeile 440 für die Zeilen-Nummer sowie der String-Tabelle zu Anfang das Werk selbst zu vollenden? Als letzter Block muß jedenfalls erscheinen:

```
960 CLS : PRINT @ 400,A;B4$
965 PRINT @ 464,J$ : PRINT @ 528 X;A1$ : PRINT @ 704,H$;C$
970 PRINT @ 832,G$ : GOTO 971
```

Falls Sie nach ähnlichen Gesichtspunkten Programme für die Umwandlung anderer Größen schreiben wollen, habe ich Ihnen in der *Tabelle 11* schon Faktoren zusammengestellt, die Sie genau so verwenden können wie F(N) im Währungsprogramm.

Tabelle 11: Faktoren für Umrechnungen von Größen

1. Längenmaße

von	in	Faktor	von	in	Faktor
Zoll (in)	Millimeter (mm)	25.4	Millimeter (mm)	Zoll (in)	.03937
Zoll (in)	Centimeter (cm)	2.54	Centimeter (cm)	Zoll (in)	.3937
Fuß (ft)	Zoll (in)	12	Zoll (in)	Fuß (ft)	.0833
Fuß (ft)	Meter (m)	.3048	Meter (m)	Fuß (ft)	3.281
Yard (yd)	Zoll (in)	36	Zoll (in)	Yard (yd)	.0278
Yard (yd)	Meter (m)	.9144	Meter (m)	Yard (yd)	1.0936
Meile (mile)	Meter (m)	1609.34	Meter (m)	Meile (mile)	.00062
Meile (mile)	Kilometer (km)	1.609	Kilometer (km)	Meile (mile)	.621
Seemeile (nmile)	Kilometer (km)	1.852	Kilometer (km)	Seemeile (nmile)	.534

2. Flächenmaße

von	in	Faktor	von	in	Faktor
Quadratzoll (in^2)	Quadratmillimeter (mm^2)	645.2	Quadratmillimeter (mm^2)	Quadratzoll (in^2)	.00155
Quadratzoll (in^2)	Quadratcentimeter (cm^2)	6.452	Quadratcentimeter (cm^2)	Quadratzoll (in^2)	.155
Quadratfuß (ft^2)	Quadratzoll (in^2)	144	Quadratzoll (in^2)	Quadratfuß (ft^2)	.00694
Acre (–)	Quadratmeter (m^2)	4046.86	Quadratmeter (m^2)	Acre (–)	.00025
Acre (–)	Hektar (ha)	.404686	Hektar (ha)	Acre (–)	2.471

3. Volumenmaße

von	in	Faktor	von	in	Faktor
Gallone (USA) (gal)	Liter (l)	3.78541	Liter (l)	Gallone (gal)	.2642
Gallone (England) (gal)	Liter (l)	4.54609	Liter (l)	Gallone (gal)	.219
Registertonne (RT)	Kubikmeter (m^3)	2.832	Kubikmeter (m^3)	Registertonne (RT)	.353
Pint (England) (pt)	Liter (l)	.56826	Liter (l)	Pint (pt)	1.76
Quart (England) (–)	Liter (l)	1.13652	Liter (l)	Quart (–)	.88
Barrel (USA) (–)	Liter (l)	158.99	Liter (l)	Barrel (–)	.0063

4. Geschwindigkeiten

von	in	Faktor	von	in	Faktor
km/h	m/s	.27778	m/s	km/h	3.6
mile/h (mph)	km/h	1.60934	km/h	mile/h (mph)	.621
Knoten (kn)	km/h	1.852	km/h	Knoten (kn)	.5399

14.4 Programme zur Datenverarbeitung

14.4.1 Sie „verwalten" ein Reitturnier – Umgang mit Listen

In der folgenden Programmserie wollen wir uns mit der „Verarbeitung" von Daten beschäftigen. Das heißt: wir wollen Listen von Daten nach wechselnden Gesichtspunkten „verarbeiten".

Als Beispiel wählen wir die
„Durchführung und Organisation einer Sportveranstaltung mit Computer-Unterstützung".
Wenn hier als Sportart eine Reitsportveranstaltung gewählt ist, dann schlicht deshalb, weil ich mich da ein wenig in den Regeln auskenne.

Wie immer kommt es mir darauf an, Ihnen vorzuführen, wie Sie ein Programm aufbauen sollten. Sie sollten in der Lage sein, das Programm soweit zu durchschauen, daß Sie es selbst für andere Fälle abwandeln können. Vielleicht wollen Sie lieber ein Tennisturnier organisieren oder die Mitgliederdaten Ihres Briefmarkenvereins „verwalten"? Sie lernen also das Prinzip.

Im vorliegenden Falle möchte ich Ihnen sogar ausdrücklich abraten, Ihr nächstes Reitturnier nach meinen Vorschlägen zu bearbeiten. Sie würden vemutlich ein Fiasko erleben. Warum ich das vermute, darf ich Ihnen natürlich nicht vorenthalten.

- Sie arbeiten bei den Programmen nur mit Tastatur-Bildschirm. In der Praxis müssen Sie aber – beispielsweise – auch Listen ausdrucken.
- Sie arbeiten für Programme und Daten nur im RAM-Bereich Ihres Computers. Können Sie sich vorstellen, was passiert, wenn in der Hitze des Gefechtes jemand den Strom abschaltet? Oder Sie bei der Eingabe über INPUT... eine einzige „falsche Bewegung" machen? Ihr Turnier wäre vermutlich gelaufen.
  In der Praxis brauchen Sie für Daten/Programme einen „nichtflüchtigen" Speicher; etwa eine Floppy. Ein Kassettenrecorder ist – weil zu umständlich und zu langsam – leider dafür nicht geeignet.

Einigen wir uns also darauf, daß wir beim „Programmieren-lernen" bleiben. Verfügen Sie – vielleicht später – über Floppy und Drucker, dann steht Ihrer Kreativität nichts mehr im Wege, aus den Programmierübungen lauffähige Programme zu machen.

Für den weiteren Fortgang wollen wir annehmen, daß Ihre Veranstaltung allen Interessierten bekannt gemacht wurde und Sie jetzt zu Hause sitzen und darauf warten, daß Ihnen der Postbote täglich ein paar „Nennungen" – von Reitern, die am Turnier teilnehmen wollen – ins Haus bringt. Bis zum „Nennungsschluß" tun Sie weiter nichts, als die Nennungen zu sammeln.

Die „Nennung" geschieht auf einem Formular, auf dem – neben Daten, die wir hier bewußt vernachlässigen – folgendes angegeben ist:

Name des Pferdes: Fury
Name des Reiters: Horst Frank
Teilnahme an den Prüfungen: 1,3,4
In unserem Beispiel soll Ihre Veranstaltung aus vier Prüfungen bestehen, etwa:
1 = einfacher Reiterwettbewerb
2 = Dressurprüfung, Klasse A
3 = Dressurprüfung, Klasse L
4 = Springprüfung, Klasse M

Stören Sie sich nicht an dem „Fachchinesisch", das es sicher für jede Sportart gibt!

Und heute ist Nennungsschluß! Sie nehmen den Stapel der eingegangenen Nennungen, setzen sich an Ihren Computer und übertragen die Daten von den einzelnen Formularen in Ihren Apparat. In dem steckt natürlich bereits Ihr Programm!

Damit wir in Zukunft wissen, wovon wir reden, will ich Ihren Stapel mit Nennungsformularen durch die folgende Tabelle ersetzen; Jede Zeile der Tabelle entspricht also dem Inhalt eines Formulars. Einverstanden, wenn wir für Übungszwecke die Liste auf zehn Teilnehmer beschränken?

Dann los!

	Pferd	Reiter	Teilnahme an Prüfung			
			1	2	3	4
1	Fury	Horst Frank	ja	–	ja	ja
2	Goldkron	Günter Süßmann	–	–	ja	ja
3	Blacky	Stefanie Simon	ja	ja	ja	ja
4	Bubi	Susanne Busch	–	ja	ja	–
5	Obelix	Werner Napp	ja	–	–	ja
6	Granada	H.-Heinrich Priehs	–	–	–	ja
7	Jenny	Hanna Wemmel	ja	–	–	–
8	Nicky	Ulla Prilop	–	ja	ja	–
9	Goldjunge	Erika Weber	ja	ja	ja	–
10	Mister X	Anna Wilke	ja	ja	ja	ja

Bevor wir uns ans Programmieren machen, wollen wir uns erst darüber verständigen, was der Computer mit den eingegebenen Daten machen soll; welche Ausgabe Sie von ihm verlangen.

- Sie wünschen eine Teilnehmerliste, die nach Pferdenamen alphabetisch sortiert ist. Gleichzeitig soll der Computer für jedes teilnehmende Pferd eine „Kopf-Nummer" erzeugen, unter der das Pferd im gesamten Verlauf des Turniers „geführt" wird. Diese Kopf-Nummer tragen dann die Pferde deutlich sichtbar mit sich herum.

- Sie wünschen eine weitere Teilnehmerliste, die jedoch nach Reiternamen sortiert ist. Diese Liste soll zusätzlich vom Computer daraufhin untersucht werden, ob ein Reitername mehr als einmal darin auftaucht.

- Schließlich soll der Computer insgesamt vier Listen bereitstellen, auf denen die Teilnehmer jeder der Prüfungen 1...4 geordnet nach Pferdenamen aufgeführt sind. Als Zusatzleistung soll der Computer dabei die Reihenfolge festlegen, nach der die Reiter an den Start gehen. Das könnte zwar in alphabetischer Reihenfolge geschehen. Sie als Veranstalter müssen sich aber an die Regeln und Vorschriften des Reiterverbandes halten. Und der hat zu Anfang des Jahres per Rundschreiben verkündet:
  ...„Am 26. Mai 19... – dem Tag Ihrer Veranstaltung – gilt für die Startfolge:

Prüfung 1 Buchstabe O
Prüfung 2 Buchstabe B
Prüfung 3 Buchstabe J
Prüfung 4 Buchstabe M

Das heißt: In Prüfung 2 startet als erstes das Pferd, dessen Name mit B beginnt; die anderen folgen in alphabetischer Reihenfolge. Also: B, C, D...Z, A

Diese Wünsche werden uns eine Weile beschäftigen. Ich möchte Ihnen jedoch vorab die „Lücken" in den folgenden Programmen aufzeigen, die ich ganz einfach deshalb lassen muß, damit aus diesem Kapitel nicht ein ganzes Buch wird.

- Wenn Sie Ihre Liste eingegeben haben, besteht keine Möglichkeit, Nachzügler anzuhängen oder Änderungen an den einmal eingegebenen Daten durchzuführen. Sie müßten dann – leider – die ganze Liste neu eingeben.

- Entdecken Sie einen Reiter zwei- oder mehrmals, dann haben Sie keine Möglichkeit, Dopplungen aus der Liste zu entfernen. Auch hier hilft ggf. nur Neueingabe der gesamten Liste.

- Sollte die Startfolge etwa mit dem Buchstaben A beginnen, dann scheitert das Programm, weil unter Ihren Nennungen kein Pferd mit dem Anfangsbuchstaben A vorkommt.

Diese „Mängelliste" enthält also drei Anregungen, die Programme zu verbessern!

Und nun beginnen wir mit dem Programmieren! Und zwar so, wie Sie das gelernt haben!

● **Titel, Auswahltabelle**

```
100 CLS
105 PRINT @ 21, "REITERVEREIN HUBERTUS"
110 PRINT"=========================================
======"
120 PRINT : PRINT
130 PRINT"DIESES PROGRAMM STELLT ZUR VERFÜGUNG:"
140 PRINT"DATENEINGABE........................ KENNZIFFER 1"
150 PRINT"TEILNEHMERLISTE PFERDE ........ KENNZIFFER 2"
160 PRINT"TEILNEHMERLISTE REITER ......... KENNZIFFER 3"
170 PRINT"TEILNEHMER PRUEFUNG 1........... KENNZIFFER 4"
180 PRINT"TEILNEHMER PRUEFUNG 2........... KENNZIFFER 5"
190 PRINT"TEILNEHMER PRUEFUNG 3........... KENNZIFFER 6"
200 PRINT"TEILNEHMER PRUEFUNG 4........... KENNZIFFER 7"
210 PRINT
220 PRINT"DRUECKEN SIE DIE GEWUENSCHTE KENNZIFFER,DANACH
    'E N T E R'"
```

Es folgen jetzt Programmauswahl und Sperre gegen Falscheingabe.

```
230 INPUT Q
240 IF Q < 1 OR Q > 7 THEN 260
250 ON Q GOTO 300,400,500,700,900,1000,1100
260 CLS : PRINT"SIE HABEN EIN NICHT EXISTIERENDES PROGRAMM AUF-
    GERUFEN!"
270 FOR X = 1 TO 1000 : NEXT : GOTO 100
```

Für die Kennziffern 4, 5, 6 und 7 haben Sie Programme vorgesehen, die nahezu identisch sind. Zur Übung begnügen wir uns jedoch mit einer Prüfung (Kennziffer 4). Deshalb überschreiben Sie noch einmal:

```
240 IF Q < 1 OR Q > 4 THEN 260
250 ON Q GOTO 300,400,500,700
```

Eingeben können wir Ihre Nennungsformulare immer noch nicht. Wir müssen noch folgende Vereinbarungen treffen:

In dem Programm werden numerische- und Stringvariable auftreten, auf deren Namen wir uns wie folgt einigen wollen:

– Start-Nummer = C
– Kopf-Nummer = KN$
– Pferd = P$
– Reiter = R$
– Prüfung 1 = N1
– Prüfung 2 = N2
– Prüfung 3 = N3
– Prüfung 4 = N4

Damit können wir nachholen:

```
10 CLEAR 1000
20 DIM C(50) : DIM KN$(50) : DIM P$(50) : DIM R$(50) : DIM N1(50)
30 DIM N2(50) : DIM N3(50) : DIM N4(50)
```

Wir fahren fort mit dem Programm für den Eingabeteil:

```
300 CLS
303 PRINT"PROGRAMM DATENEINGABE:"
305 PRINT"=====================================
======"
310 PRINT"BITTE GEBEN SIE AUF ANFORDERUNG DIE DATEN EIN!"
320 PRINT"BITTE BEACHTEN SIE DABEI:"
325 PRINT"NAME DES REITERS IN DER FORM : MEIER WILLI"
330 PRINT"FUER TEILNAHME AN EINER PRÜFUNG                1 EINGEBEN"
332 PRINT"FUER NICHT-TEILNAHME                           0 EINGEBEN"
334 PRINT"IST DIE LISTE ZUENDE, GEBEN SIE 'E N D E' ANSTELLE DES
    PFERDENAMENS EIN !": PRINT
336 PRINT"WENN SIE BEREIT SIND, DRUECKEN SIE 'E N T E R'!"
338 INPUT Z

340 CLS : FOR I = 1 TO 50 : CLS
```

Und nun programmieren wir das „Generieren" der Kopfnummer. Dazu verwandeln wir die Laufnummer I der FOR..NEXT..-Schleife in eine indizierte Stringvariable. Durch die Zeile:

```
341 KN$(I) = STR$(I)
```

Wir fahren fort:

```
342 PRINT"EINGEBEN FÜR KOPF-NUMMER";KN$(I)
343 PRINT"*****************************************************
*********"
344 INPUT"NAME DES PFERDES";P$(I)
345 IF P$(I)="ENDE" THEN N=I-1 : GOTO 360
346 INPUT"NAME DES REITERS";R$(I)
348 INPUT"PRUEFUNG 1 :";N1(I)
350 INPUT"PRUEFUNG 2 :";N2(I)
352 INPUT"PRUEFUNG 3 :";N3(I)
354 INPUT"PRUEFUNG 4 :";N4(I)
356 NEXT I
360 CLS : PRINT"DIE LISTE IST ABGESCHLOSSEN!"
365 PRINT"SIE HABEN ";N;" TEILNEHMER EINGEGEBEN!"
370 FOR X = 1 TO 1000 : NEXT : GOTO 100
```

Bemerken Sie, daß Zeile 370 das Programm wieder ins Menü schickt? Aus gutem Grund! Würden wir das nicht tun, würde das Programm in 365 beendet sein und der Computer in den READY-Zustand zurückkehren. Damit wären alle über INPUT... eingegebenen Daten wieder gelöscht! Sie müssen also bei Programmen dieser oder ähnlicher Art verhindern, daß das Programm zum Ende kommt! Es sei denn, Sie wünschen das ausdrücklich so.

Als Zwischenspiel setze ich das Programm wie folgt fort:

```
370 FOR X = 1 TO 1000 : NEXT : CLS
380 FOR I = 1 TO N
390 PRINT KN$(I),P$(I),R$(I)
400 NEXT
410 PRINT KN$(7),P$(7),R$(7)
```

Geben Sie nun die Daten unserer vereinbarten Teilnehmerliste ein. Sie erhalten sie so ausgegeben, wie eingegeben. Beachten Sie die Programmzeile 410! Die habe ich zum Beweis dafür eingebaut, daß wir tatsächlich aus der Laufnummer I eine indizierte Stringvariable gemacht haben! Wie würden Sie sonst die 7. Zeile der Eingabe wiederfinden?

Bevor wir mit unserem Programm weitermachen, überschreiben Sie:

```
370 FOR X = 1 TO 1000 : NEXT : GOTO 100
```

und löschen Sie die Zeile 380, 390, 400 und 410!

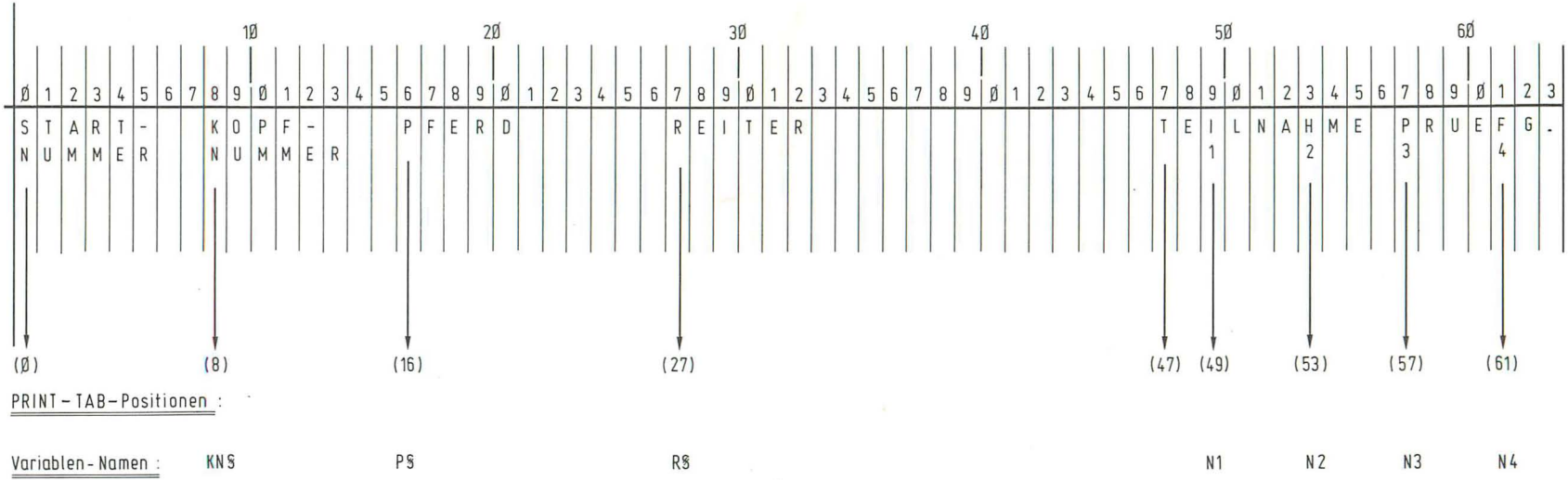


Abb. 41 Hilfe für die „Konstruktion“ der Kopfzeile

● Teilnehmerliste

Die erste gewünschte Liste erfordert ein Sortieren nach Pferdenamen. Das Sortierprogramm wollen wir als Unterprogramm entwerfen. Da das Sortieren eine gewisse Zeit in Anspruch nimmt, wollen wir dem Operateur sofort die Kopfzeile liefern, damit er nicht denkt, der Computer würde Pause machen!

Für die „Konstruktion" der Tabelle zeige ich Ihnen die *Abb. 41*. Da mehr als vier Spalten auszugeben sind, ist eine solche Darstellung sehr hilfreich.

```
400 CLS : PRINT TAB(0)"START-";TAB(8)"KOPF-";TAB(16)"PFERD";TAB-
    (27)"REITER";TAB(47)"TEILNAHME PRUEFG.";TAB(0)"NUMMER";TAB-
    (8)"NUMMER";TAB(49)"1";TAB(53)"2";TAB(57)"3";TAB(61)"4"
410 PRINT"=========================================
======"
```

Ins Sortierprogramm geht's mit:

```
420 GOSUB 1000
```

Und jetzt folgt bereits der Programm-Text für die Ausgabe nach Rückkehr aus dem Unterprogramm:

```
430 FOR I = 1 TO N
440 PRINT TAB(8)KN$(I);TAB(16)P$(I);TAB(27)R$(I);TAB(49)N1(I);TAB(53)N2
    (I);TAB(57)N3(I);TAB(61)N4(I)
450 NEXT I
```

Damit ist das Programm für die erste Liste fertig. Sie können das Programm durch Druck auf ENTER wieder ins Menü befördern. Etwa so:

```
460 INPUT Z : GOTO 100
```

Das ist für ein derartiges Programm sicher sinnvoll, weil es die eingegebenen Daten nicht verliert. Sie können aber auch – sicher selbständig – den Benutzer fragen:

– ob er weiterarbeiten will... dann zurück ins Menü.
– ob er genug hat... dann zu 2000 END

Es folgt jetzt das Sortierprogramm. Vergleichen Sie es mit dem genau gleichen – was die Wirkung angeht! – auf Seite 140. Und merken Sie sich:
– Bei der Übernahme von fremden Programmen/Routinen auf Anzahl, Art und Namen der verwendeten Variablen achten.

- Achten Sie darauf, wieviele Umstellungen zu programmieren sind! Betroffen sind die Zeilen
  1130...1134
  1170...1174
  1230...1234.
- Beachten Sie wegen der anderen Zeilen-Numerierung die geänderten Ziele für die GOTOs!

Also: Programme nie wörtlich; immer sinngemäß übernehmen!

```
1000 L = 1
1010 B(L)=N+1
1020 M = 1
1030 J = B(L)
1040 I = M - 1
1050 IF J - M < 3 THEN 1210
1060 M1 = INT(I+J)/2)
1070 I = I + 1
1080 IF I = J THEN 1150
1090 IF P$(I)<P$(M1) THEN 1070
1100 J = J - 1
1110 IF I = J THEN 1150
1120 IF P$(J) > = P$(M1) THEN 1100
1130 P$(0)=P$(I) : KN$(0)=KN$(I) : R$(0)=R$(I) : N1(0)=N1(I) : N2(0)=N2(I) : N3
     (0)=N3(I) : N4(0)=N4(I)
1132 P$(I)=P$(J) : KN$(I)=KN$(J) : R$(I)=R$(J) : N1(I)=N1(J) : N2(I)=N2(J) :
     N3(I)=N3(J) : N4(I)=N4(J)
1134 P$(J)=P$(0) : KN$(J)=KN$(0) : R$(J)=R$(0) : N1(J)=N1(0) : N2(J)=N2(0) :
     N3(J)=N3(0) : N4(J)=N4(0)
1140 GOTO 1070
1150 IF I > = M1 THEN I = I - 1
1160 IF J = M1 THEN 1180
1170 P$(0)=P$(I) : KN$(0)=KN$(I) : R$(0)=R$(I) : N1(0)=N1(I) : N2(0)=N2(I) : N3
     (0)=N3(I) : N4(0)=N4(I)
1172 P$(I)=P$(M1) : KN$(I)=KN$(M1) : R$(I)=R$(M1) : N1(I)=N1(M1) :
     N2(I)=N2(M1) : N3(I)=N3(M1) : N4(I)=N4(M1)
1174 P$(M1)=P$(0) : KN$(M1)=KN$(0) : R$(M1)=R$(0) : N1(M1)=N1(0) :
     N2(M1)=N2(0) : N3(M1)=N3(0) : N4(M1)=N4(0)
1180 L = L + 1
1190 B(L)=I
1200 GOTO 1030
1210 IF J - M < 2 THEN 1240
```

```
1220 IF P$(M) < P$(M + 1) THEN 1240
1230 P$(0)=P$(M):KN$(0)=KN$(M):R$(0)=R$(M):N1(0)=N1(M):N2(0)=N2(M):
     N3(0)=N3(M) : N4(0)=N4(M)
1232 P$(M)=P$(M+1):KN$(M)=KN$(M+1):R$(M)=R$(M+1):N1(M)=N1(M+1):
     N2(M)=N2(M+1) : N3(M)=N3(M+1) : N4(M)=N4(M+1)
1234 P$(M+1)=P$(0) : KN$(M+1)=KN$(0) : R$(M+1)=R$(0) : N1(M+1)=N1(0) :
     N2(M+1)=N2(0) : N3(M+1)=N3(0) : N4(M+1)=N4(0)
1240 M=B(L)+1
1250 L=L-1
1260 IF L > 0 THEN 1030

1270 RETURN
```

● Noch einmal: Teilnehmerliste

Die nächste Liste soll zunächst nur nach den Reitern sortiert werden. Für das Sortierprogramm bedeutet das:
– In Zeile 1090 – z. B. – wird nicht mehr P$(I) mit P$(M1) verglichen...
– Es muß R$(I) mit R$(M1) verglichen werden.
Soll das etwa heißen, das das ellenlange Sortierprogramm oben noch einmal einzugeben ist?

Nein. Wir schlagen das Programm gewissermaßen mit seinen eigenen Waffen. Es werden einfach vor dem Sortierprogramm die beiden Strings P$ und R$ miteinander vertauscht. Das geschieht unten in den Zeilen 520...540.

Zuvor soll der Ausgabeteil analog zum vorigen Programm eingegeben werden; wegen der geänderten Tabelle sind andere TABs notwendig.

```
500 CLS : PRINT TAB(0)"START-";TAB(8)"KOPF-";TAB(16)"REITER";TAB
    (36)"PFERD";TAB(47),,TEILNAHME PRUEFG.";TAB(0)"NUMMER";TAB
    (8)"NUMMER";TAB(49)"1";TAB(52)"2";TAB(57)"3";TAB(61)"4"
510 PRINT"=========================================
======"

515 FOR I = 1 TO N

520 H$=P$(I)
530 P$(I)=R$(I)
540 R$(I)=H$

550 NEXT I

555 GOSUB 1000
```

Wenn Sie aus dem Unterprogramm wieder auftauchen, müssen Sie die in den Zeilen 52Ø...54Ø betriebene Vertauschung wieder rückgängig machen. Weil andernfalls bei einem nochmaligen Aufruf der Kennziffer 3 in den Zeilen 52Ø ...54Ø diese Rückverwandlung automatisch geschehen würde und in der Ausgabe Roß und Reiter verwechselt würden. Also:

```
56Ø FOR I = 1 TO N

57Ø H$=R$(I)
58Ø R$(I)=P$(I)
59Ø P$(I)=H$

595 PRINT TAB(8)KN$(I);TAB(16)R$(I);TAB(36)P$(I);TAB(49)N1(I);TAB(53)N2
    (I);TAB(57)N3(I);TAB(61)N4(I)
597 NEXT I
```

Das Ende des dritten Programmteiles können Sie genau so abschließen wie das des zweiten; also etwa:

```
6ØØ INPUT Z : GOTO 1ØØ
```

● Die Liste wird auf Mehrfach-Nennungen durchsucht

Wenden wir uns nun dem zweiten Teil der Aufgabe zu. Wie Sie sich erinnern werden, soll die inzwischen nach Reiternamen geordnete Liste daraufhin untersucht werden, ob ein (oder mehrere) Reiternamen darin öfter als einmal vorkommen. Diese Durchsuchung legen wir bewußt an das Ende des Sortiervorganges; stehen doch etwaige Mehrfach-Nennungen nach dem Sortieren hübsch untereinander.

Der Übersichtlichkeit halber soll diese Teilaufgabe isoliert behandelt werden. Im folgenden wird dazu ein neuer Eingabeteil entworfen. Zur Schonung Ihrer Tastatur einigen wir uns auf die ebenfalls folgende „Namensliste".

Diese Art der Programmierung empfehle ich Ihnen generell für den Entwurf komplexerer Programme; ohne viel Tipparbeit können Sie so leicht prüfen, ob Ihr Teilproblem sauber gelöst wird. Ist das der Fall, können Sie das entsprechende Programm leicht selbst in das Hauptprogramm einbinden. Wollen Sie dieses „retten", dann beginnen Sie das Teilprogramm gleich mit der Zeilennummer 2ØØ5 (statt 5) und starten die Zwischenübungen mit RUN 2ØØ5. Kann's losgehen?

1. Teil: Eingabe

```
 5 CLS
10 CLEAR 1000
20 DIM R$(50)
30 FOR I = 1 TO 50
40 INPUT"NAME =";R$(I)
50 IF R$(I) = "ENDE" THEN N = I - 1 : GOTO 80
70 NEXT I
80 PRINT"ENDE DER EINGABE VON ";N;" TEILNEHMERN"
85 FOR X = 1 TO 1000 : NEXT
```

Als „Namensliste" für die Eingabe bei Zwischentests wählen wir:

A
B
B
C
D
E
E
E
F
G
ENDE

Sie sehen, daß die „Namensliste" bereits alphabetisch sortiert vorliegt.

1. Versuch

```
100 FOR I = 1 TO N
110 IF R$(I)=R$(I+1) THEN PRINT I,R$(I)
120 NEXT I
```

In Zeile 110 ist also eine „Suchformel" programmiert, die feststellen soll, ob der aktuelle Name in der FOR..NEXT..-Schleife identisch ist mit dem folgenden.

Das Ergebnis ist:

2	B
6	E
7	E

Mit der Formel in 11Ø erkennen wir also nicht den zweiten Namen B bzw. den dritten Namen E – was, wenn Sie darüber nachdenken – auch nicht verwunderlich ist. Ändern Sie:

```
11Ø IF R$(I) = R$(I+1) OR R$(I) = R$(I−1) THEN PRINT I,R$(I)
```

Jetzt erhalten Sie richtig:

```
2        B
3        B
6        E
7        E
8        E
```

Der erste Teil der geänderten Suchformel erkennt also den 2., 6. und 7. Namen; der zweite Teil den 3. und den 8.

Für die endgültige Ausgabe wählen wir:

```
1ØØ CLS : PRINT"FOLGENDE REITER SIND MEHRFACH GENANNT:"
1Ø5 PRINT"=====================================
======"
11Ø FOR I = 1 TO N
12Ø IF R$(I)=R$(I+1) OR R$(I)=R$(I−1) THEN PRINT I,R$(I)
13Ø NEXT I
```

Die Einbindung in das Hauptprogramm können Sie sicher selbst vornehmen – falls Sie das überhaupt wünschen. Denken Sie ggf. an eine Rückkehr ins Menü durch:

```
15Ø INPUT Y : GOTO 1ØØ
```

Erlauben Sie mir zu dem letzten Programm noch einige Hinweise:

- Die „Suchformel" in Zeile 12Ø funktioniert nur mit der vorgeschlagenen Form der Dateneingabe (Zeilen 5...85)
- Denn: in dem letzten Lauf der FOR...NEXT..-Schleife „befehlen" Sie: ...IF R$(1Ø) = R$(11) OR...
- Sie haben tatsächlich elf Namen eingegeben; der 11. war „ENDE"!
- Beginnen Sie Ihre Eingabe etwa mit N = 1Ø – weil Sie die Anzahl der Namen schon kennen – so erhalten Sie nach der Suchformel von den meisten Computern eine Fehlermeldung.

● Wer nimmt teil an Prüfung 1?

Mit der Kennziffer 4 läßt sich bekanntich abrufen, welcher Reiter bzw. welches Pferd an Prüfung 1 teilnimmt. Das entsprechende Programm müßte in Zeile 7ØØ

beginnen. Auch bei dieser Aufgabe wollen wir uns auf das Detail konzentrieren und verlassen dazu das Hauptprogramm. Denn es wird ganz schön knifflig! Unter der Kennziffer 4 sind folgende Jobs zu erledigen:

A) Die eingegebene Liste ist nach Pferdenamen zu sortieren.

B) Es ist festzustellen, welche Pferde/Reiter an der Prüfung 1 teilnehmen.

C) Aus der so „verdichteten" Teilnehmerliste ist das Pferd mit dem Anfangsbuchstaben „O" in seinem Namen zu suchen.

D) Für dieses ist die Start-Nummer 1 festzulegen, da es nach den Regeln als erstes starten soll.

E) Schließlich ist die Liste der Teilnehmer an Prüfung 1 auszudrucken; im Beispiel beginnend mit:
   1 OBELIX...
   2 BLACKY...
   3 GOLDJUNGE...
   usw.

Da wir das Sortieren bereits eingeübt haben, beschäftigen wir uns nur noch mit den Detailaufgaben B...E; der besseren Übersicht halber wollen wir sie einzeln angeben. Wieder entwerfen wir zunächst einen Eingabeteil:

```
 5 CLS
1Ø CLEAR 1ØØØ
2Ø DIM N$(5Ø) : DIM N1(5Ø)
3Ø FOR I = 1 TO 5Ø
4Ø INPUT "NAME=";N$(I)
5Ø IF N$(I)="ENDE" THEN N = I -1 : GOTO 8Ø
6Ø INPUT"TEILNAHME ? JA = 1 , NEIN = Ø";N1(I)
7Ø NEXT I
8Ø PRINT"ENDE DER LISTE VON ";N" TEILNEHMERN"
85 FOR X = 1 TO 1ØØØ : NEXT
```

Als Eingabe für die Zwischenübungen einigen wir uns auf folgende „Namensliste":

```
NAME     TEILNAHME
==============
A        Ø
B        Ø
C        1
D        Ø
E        1
F        1
G        Ø
H        1
ENDE
```

Die Liste ist also bereits wieder alphabetisch sortiert und umfaßt nur acht Namen.

● *Ausgabe der Eingabe*

Lassen Sie sich zur Überprüfung von Programmen und Eingabe die Liste so ausgeben, wie sie oben eingegeben wurde; und zwar mit:

```
100 CLS : PRINT"NAME","TEILNAHME"
110 PRINT"=====================================
======"
120 FOR I = 1 TO N
130 PRINT N$(I), N1(I)
140 NEXT I
```

Stimmt noch alles?

Im folgenden schreibe ich Ihnen jeweils das ganze Ausgabeprogramm hin, damit Sie den Überblick behalten. Sollten Sie an dieser Programmserie „hintereinanderweg" arbeiten, dann brauchen Sie jedoch nur die Programmzeilen zu überschreiben bzw. hinzuzufügen, die ich vor der Zeilennummer mit einem Punkt markiere.

Ausgabe: Wer nimmt teil an Prüfung 1?

Das erfahren Sie ganz einfach so:

```
  100 CLS : PRINT"NAME","TEILNAHME"
  110 PRINT"=====================================
  ======"
  120 FOR I = 1 TO N
● 130 IF N1(I) = 1 THEN PRINT N$(I),N1(I)
  140 NEXT I
```

Wie Sie sehen, brauchen Sie nur die Zeile 130 wie angegeben zu ändern. Sie erhalten folgende Liste:

```
NAME      TEILNAHME
==============
C         1
E         1
F         1
H         1
```

Sie haben zwar alle Teilnehmer der Prüfung 1. Aber es fehlt noch etwas!

Ausgabe: Welcher Name beginnt mit dem Buchstaben „E"?

Auch das läßt sich – für sich allein – ganz einfach herausfinden mit:

```
100 CLS : PRINT"NAME","TEILNAHME"
110 PRINT"==================================
======"
120 FOR I = 1 TO N
● 130 IF N$(I) = "E" THEN 200
140 NEXT I
```

In Zeile 130 durchsuchen wir also die Liste nach dem Anfangsbuchstaben „E".

Sollten Sie eine Liste mit „richtigen" Namen haben; suchen Sie darin etwa „EMIL", machen Sie die Suchformel zu ...IF LEFT$(N$(I),1) = "E"... Ist doch klar?

Für das spätere Aufschreiben der Liste müssen wir uns noch merken, auf welcher Position – bei welchem I – wir „E" gefunden haben.

Dazu setzen wir einen Programmiertrick ein, den Sie schon aus der Eingabe kennen. Wir sagen:

```
● 200 A = I
```

Da in unserer Eingabeliste „E" an 5. Stelle steht, wird A zu 5.

Für das Hinschreiben der Liste fahren wir fort:

```
● 210 FOR S = A TO N
● 220 PRINT N$(S),N1(S)
● 230 NEXT S
```

Das heißt im Klartext: Die Liste ist von der 5. bis zur N-ten Position hinzuschreiben; das ist hier: von E...H

Für den zweiten Teil der Liste heißt es dann:

```
● 250 FOR S = 1 TO A - 1
● 260 PRINT N$(S),N1(S)
● 270 NEXT S
```

Was das im Klartext heißt, können Sie wohl selbst übersetzen?

Das Ergebnis wird jedenfalls sein:

```
NAME      TEILNAHME
==========
E         1
F         1
G         Ø
H         1
A         Ø
B         Ø
C         1
D         Ø
```

Das ist zwar noch immer nicht ganz das, was wir wollen. Aber haben Sie noch Geduld!

● **Wir erzeugen die Start-Nummer!**

Bauen wir die Lösung für diese Detailaufgabe zunächst in das letzte Programm ein!

```
● 1ØØ CLS : PRINT"ST.-NR.-","NAME","TEILNAHME"
  11Ø PRINT"==================================
  ======"
  12Ø FOR I = 1 TO N
  13Ø IF N$(I)="E" THEN 2ØØ
  14Ø NEXT I

● 2ØØ A = I : B = Ø

  21Ø FOR S = A TO N
● 215 B = B + 1
● 22Ø PRINT B,N$(S),N1(S)
  23Ø NEXT S

  25Ø FOR S = 1 TO A - 1
● 255 B = B + 1

● 26Ø PRINT B, N$(S),N1(S)
  27Ø NEXT S
```

Sie erhalten folgende Liste:

ST.-NR.-	NAME	TEILNAHME
1	E	1
2	F	1
3	G	Ø
4	H	1
5	A	Ø
6	B	Ø
7	C	1
8	D	Ø

● **Wir fügen die Listen zusammen!**

Im letzten Teil unserer Übungen wollen wir versuchen, aus den bisher richtig gelösten Details das zu machen, was wir eigentlich haben wollen: Eine Liste, die mit „E" = 1 beginnt und nur die Teilnehmer an der Prüfung 1 enthält. Auf welche Schwierigkeiten wir dabei stoßen, werde ich Ihnen rechtzeitig sagen!

Zum Verständnis des Programmes bedienen Sie sich des Ablaufplanes auf Seite 199!

```
100 PRINT"ST.-NR.-","NAME","TEILNAHME"
110 PRINT"=================================
======"
● 115 B = Ø
120 FOR I = 1 TO N
● 130 IF N1(I)=1 THEN 150
140 NEXT I
● 150 B = B + 1
```

Bis hierher waren wir ja schon! Wir haben das alte Programm – das die „1" herausgesucht hatte, nur insoweit abgeändert, als wir jetzt mit B die N1(I) = 1 zählen; wir müßten inzwischen bei B = 4 sein.

Ja, und vorhin hatten wir jetzt programmiert, die Liste hinzuschreiben. Das wollen wir aber nicht! Wir müssen die gefundenen Namen „im Sinn" behalten, um sie gleich anschließend weiter zu bearbeiten.

„Im Sinn" – oder im Speicher – behalten kann sie doch der Computer!

Dem bringen wir das bei mit folgenden Zeilen:

```
● 151 N1$(B) = N$(I)
● 154 N2(B) = N1(I)
● 160 NEXT I
```

Wie Sie sehen, haben wir also alle Namen, die bei N1(I) eine 1 hatten, unter einem neuen Variablennamen im Speicher hinterlegt. Und mit denen arbeiten wir jetzt weiter!

```
● 170 FOR S = 1 TO B
● 180 IF N1$(S) = "E" THEN 200
● 190 NEXT S
```

Wir durchsuchen also die „Liste im Sinn" nach „E"; in der Liste stehen jetzt nur noch die vier Namen von 1...B.

```
● 200 A = S : C = 0
● 210 FOR S = A TO B
● 215 C = C + 1
● 220 PRINT C,N1$(S),N2(S)
  230 NEXT S
  250 FOR S = 1 TO A -1
● 255 C = C + 1
● 260 PRINT C,N1$(S), N2(S)
  270 NEXT S
```

Das Ergebnis des Programms ist endlich die gewünschte Liste:

```
ST.-NR.-     NAME      TEILNAHME
=======================
1            E         1
2            F         1
3            H         1
4            C         1
```

Und nun haben Sie die dankbare Aufgabe, das Ergebnis Ihrer Mühen in das „große" Programm einzubauen! Auch für diese Übung bin ich noch ein paar Bemerkungen schuldig.

- Daß ich in den Teilprogrammen gelegentlich die Namen für Zähler und Indices geändert habe, entspricht nicht sachlicher Notwendigkeit. Es geschah aus pädagogischen Gründen, damit sie lernen, daß es im Grunde gleich ist, wie Sie eine (String-)Variable benennen. Sie müssen nur innerhalb einer Programmsequenz dabei bleiben.
- Auf den Ausdruck der Spalte „TEILNAHME" hätten Sie gut verzichten können. Besonders dann, wenn Sie Ihre Liste mit der Überschrift geschmückt hätten:
  TEILNEHMER AN PRUEFUNG 1:
  ==========

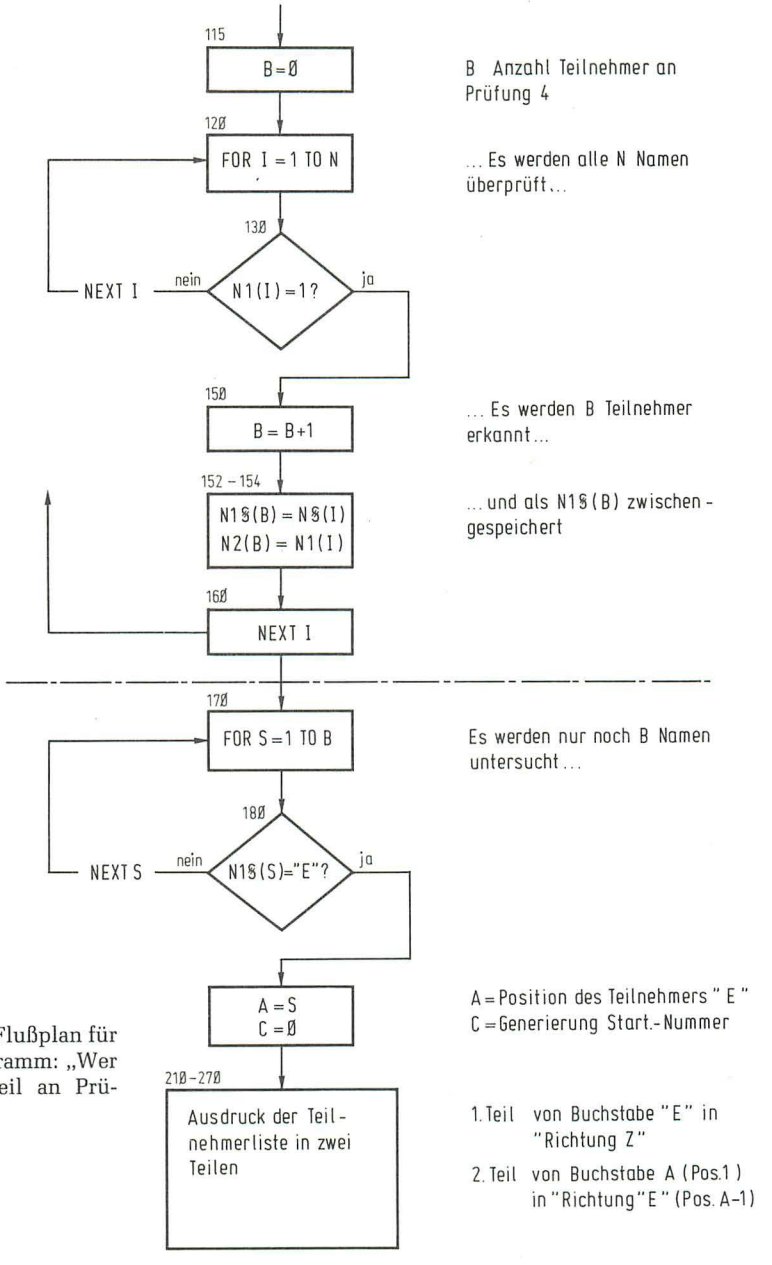


Abb. 42 Flußplan für das Programm: „Wer nimmt teil an Prüfung 1?"

– Das Programm erhält in *einer* FOR..NEXT..-Schleife (Zeile 12Ø) *zweimal* ..NEXT (Zeilen 14Ø, 16Ø). Das ist nicht ganz korrektes BASIC. Mein Computer hat das zwar toleriert. Bei anderen kann die Fehlermeldung auftauchen:

? NF ERROR... (das heißt: NEXT ohne FOR). Sie können diese Klippe nach folgendem Programmschema umschiffen:

```
10 FOR X = 1 TO 10
20 IF X = 5 THEN GOSUB 100
30 PRINT X
40 NEXT X
50 END
100 PRINT"X = 5"
110 X = X + 1
120 RETURN
```

In obigem Beispiel erhalten Sie

```
1
2
3
4
X = 5
6
7
8
9
10
```

Ihre Klippe mit „zwei mal NEXT" ist umschifft. Die *Abb. 42* zeigt den Flußplan für das letzte Programm.

● Sie ermitteln den Sieger eines Springturniers

Für die nächste Programmfolge wollen wir die Szene wechseln: Sie sitzen nun mit Ihrem Computer im Richterturm und beobachten ein Springturnier. Ihre Aufgabe wird es sein, ein Programm für die Ermittlung des Siegers und der Plazierten zu entwickeln.

Um Ihnen etwas Abwechslung zu verschaffen, wollen wir die relevanten Daten über READ..DATA.. eingeben, – statt wie bisher, mit INPUT.. zu arbeiten. Das hat zunächst bei der Programmentwicklung den Vorteil, daß die Liste der eingegebenen Daten gewissermaßen Bestandteil des Programms wird und nicht für jeden Probelauf neu eingegeben werden muß. Ein Nachteil besteht darin, daß die Handhabung des Programms im Laufe des Turniers etwas höhere Ansprüche an den Bediener stellt und eine Bedienerführung – wie in den vorigen Beispielen – fehlt. Aber Sie sind ja inzwischen (fast) Profi!

Beschäftigen wir uns zunächst mit den Randbedingungen für das zu entwikkelnde Programm.

Sie sitzen also im Richterturm und haben für den ersten Teilnehmer – der gleich einreiten wird – folgende Vorbereitung getroffen:

```
2000 DATA
```

Der Reiter erscheint und der Stadionsprecher stellt ihn vor: ...„Es startet mit der Start-Nummer 67 OBELIX unter Werner Napp..."

Sie ergänzen also Ihre Eingabe wie folgt:

```
2000 DATA 67, "OBELIX","WERNER NAPP"
```

Nun dürfen Sie zuschauen, was der Reiter im Parcour anstellt. Hat er seine Prüfung beendet, sagt Ihnen einer der Preisrichter z. B.: „8 Fehler in 65 Sekunden". Das führt bei Ihnen nun zu der endgültigen Datenzeile:

```
2000 DATA 67,"OBELIX","WERNER NAPP",8,65
```

Und während Herr Napp ausreitet und der nächste Teilnehmer erwartet wird, beschäftigen Sie sich mit Ihrem Computer:

```
2010 DATA
```

Sie erkennen, wie Sie Ihren Job zu bewältigen haben?

Nach einer Weile erscheint der fünfte Teilnehmer; Sie geben nach der Vorstellung ein:

```
2040 DATA 23,"GRANADA","ERNST WILKE"
```

Leider sind Granada und Herr Wilke heute morgen nicht einig. Granada verweigert dreimal den Gehorsam vor einem Sprung. Das ist nach den Spielregeln ein Grund, den Teilnehmer für diese Prüfung auszuschließen.

Wie sagen Sie's Ihrem Kinde; Ihrem Computer?

Wir wollen miteinander vereinbaren: Bei Ausschluß eines Teilnehmers geben Sie ein: für Fehler 999, für Zeit 0

Ihre oben begonnene Zeile lautet also vollständig:

```
2040 DATA 23,"GRANADA","ERNST WILKE",999,0
```

Und noch eine Verabredung wollen wir treffen: Hat der letzte Reiter der Prüfung den Parcour verlassen, geben Sie eine Abschluß-Zeile in Ihre Datenliste wie folgt ein:

....DATA 9999,"9999","9999"9999,9999

Für unsere weitere Programmentwicklung wollen wir von 15 Teilnehmern ausgehen; bitte geben Sie die DATA-Liste doch jetzt einmal in Ihren Computer ein (*Tabelle 12*).

Tabelle 12: Die Teilnehmer am Springturnier

```
2000 DATA 67,"OBELIX","WERNER NAPP",8,65
2010 DATA 120,"NICKY","HEINRICH PRILOP",0,58
2020 DATA 15,"WALDESLUST","SUSANNE BUSCH",0,55
2030 DATA 7,"ELFIE","WILLI SOWA"12,67

2040 DATA 23,"GRANADA","ERNST WILKE",999,0
2050 DATA 1,"GO-ON","OTTO GRUSSENDORF",8,63
2060 DATA 9,"BLACKY","STEFANIE SIMON",4,60
2070 DATA 170,"SIMONA","WOLFGANG SIMON",4,62
2080 DATA 18,"KULKI","KATJA WILKE",999,0
2090 DATA 35,"JOKER","SIGRID GUTHY",0,60
2100 DATA 12,"FANTANELLA","ERNST MEIER",999,0
2110 DATA 17,"GOLDKRONE","GUENTHER SUESSMANN",0,50
2120 DATA 43,"LUMPI","HEINRICH PRIEHS",0,52
2130 DATA 19,"ROSI","HANNA WEMMEL",999,0
2140 DATA 12,"KOBOLD","JOERN MERZDORF",4,61
2150 DATA 9999,"9999","9999",9999,9999
```

Namen für die Variablen:

START-NUMMER	= S(N)
PFERD	= P$(N)
REITER	= R$(N)
FEHLER	= F(N)
ZEIT	= T(N)

Und nun zum Einleseprogramm für unsere Datenliste! Da Sie im voraus nicht wissen können, ob 15 Reiter teilnehmen oder 150, muß Ihr Programm wieder „allgemeingültig" werden. Das heißt, der Computer soll erst einmal zählen, wieviele Teilnehmer Sie überhaupt eingegeben haben. Das geschieht unten in den Zeilen 5, 15 und 40. Es geht los!

```
  3 CLS : CLEAR 1ØØØ
  5 N = Ø
 1Ø READ S,P$,R$,F,T
 15 N=N+1
 2Ø IF S = 9999 THEN 4Ø
 3Ø GO TO 1Ø
 4Ø N = N-1

1ØØ DIM S(N),P$(N),R$(N),F(N),T(N)
```

Da wir erst jetzt – in Zeile 1ØØ – die DIMensionierung vornehmen, wird nur soviel Speicherplatz belegt, wie tatsächlich gebraucht wird! Wir entwerfen nun die Überschrift für die auszugebende Liste:

```
25Ø PRINT   TAB(Ø)"PLATZ-";TAB(8)"START-";TAB(16)"PFERD";TAB(28)"REI-
    TER";TAB(47)"FEHLER";TAB(55)"ZEIT"
26Ø PRINT TAB(Ø)"NUMMER";TAB(8)"NUMMER"
27Ø PRINT"=====================================
======"
```

In einem ersten Probelauf testen wir, ob die Liste so ausgegeben wird, wie sie in der DATA-Liste eingegeben wurde. Dabei ist Zeile 255 besonders wichtig!

```
255 RESTORE
3ØØ FOR I = 1 TO N
31Ø READ S(I),P$(I),R$(I),F(I),T(I)
32Ø PRINT TAB(Ø)I;TAB(8)S(I);TAB(16)P$(I);TAB(28)R$(I);TAB(47)F(I);TAB
    (55)T(I)
33Ø NEXT I
34Ø PRINT P$(1Ø),,,R$(1Ø)
```

Durch den Ausdruck von Zeile 34Ø vergewissern wir uns, daß die Daten richtig indiziert worden sind.

Falls Sie mit dem Probelauf zufrieden sind, fahren Sie fort:

```
312 NEXT I
314 GOSUB 1ØØØ
316 FOR I = 1 TO N
```

Löschen Sie Zeile 34Ø!

Im Unterprogramm – das in Zeile 1000 beginnt – wird nun die Namensliste sortiert. Dabei dient als Sortierkriterium die Fehleranzahl, weil nach den Regeln der Reiter mit der geringsten Fehlerzahl gewinnt.

```
1000 R = N - 1
1010 FOR Z = 1 TO N - 1
1020 FOR Y = 1 TO R
1030 IF F(Z) < F(Z+Y) THEN 1070
1040 H = S(Z):H$=P$(Z):H1$=R$(Z):H1=F(Z):H2=T(Z)
1050 S(Z)=S(Z+Y):P$(Z)=P$(Z+Y):R$(Z)=R$(Z+Y):F(Z)=F(Z+Y):T(Z)=T(Z+Y)
1060 S(Z+Y)=H:P$(Z+Y)=H$:R$(Z+Y)=H1$: F(Z+Y)=H1:T(Z+Y)=H2
1070 NEXT Y
1080 R = R - 1
1090 NEXT Z
1100 RETURN
```

Falls Sie jetzt einen Probelauf machen, werden Sie erkennen, daß die vier ausgeschiedenen Reiter – das waren nach unserer Vereinbarung die Reiter, deren Fehlernummer mit 999 angegeben war – am Ende der Liste erscheinen. Da die Reiter ausgeschieden sind, brauchen wir sie für die weitere Bearbeitung auch nicht mehr; wir können Sie aus der Liste ausscheiden. Dazu ermitteln wir die Anzahl derjenigen Reiter, die noch zu werten sind; in unserem Beispiel sind das noch 11.

Fügen Sie dazu in das Programm ein:

```
272 W = N
274 FOR I = 1 TO N
276 READ S(I),P$(I),R$(I),F(I),T(I)
278 IF F(I) = 999 THEN 284
280 NEXT I
282 RESTORE : GO TO 300
284 W = W -1
286 GO TO 280
```

Für den Ausdruck überschreiben wir noch:

```
316 FOR I = 1 TO W
```

Ferner – zur Vermeidung einer Fehlermeldung –:

```
340 END
```

Ein erneuter Probelauf zeigt nun:
- Die Liste enthält nur noch die Reiter, die in die Wertung kommen.
- Die Liste ist nach aufsteigender Fehlerzahl sortiert.

Leider können wir mit der Liste noch nichts anfangen, da sie nicht den Regeln entspricht. Die lauten nämlich:

- Es gewinnt der Reiter mit der geringsten Fehlerzahl.
- Bei zwei – oder mehreren – Reitern mit *gleicher Fehlerzahl* gewinnt der Reiter, der den Parcour in der kürzesten Zeit absolviert hat.

Sie erkennen, daß genau diese Bedingung von unserer Liste nicht erfüllt wird.

● Sie sortieren in einer sortierten Liste!

Zur Lösung des noch verbleibenden Problems trennen wir uns von der vorigen Aufgabe und beginnen eine neue Programmfolge, die an unsere Überlegungen in Kapitel 3, Seite 15 anknüpft.

Unser Springturnier hat doch offensichtlich drei mögliche Ergebnisse, die ich Ihnen einmal für zehn Teilnehmer aufzeichne. Dabei sind nur die hier interessierenden Variablen F = Fehler und T = Zeit aufgeführt.

Möglichkeit 1

Alle Reiter haben den Parcour mit der *gleichen Fehlerzahl* absolviert. In diesem Falle ist die Liste nur nach der *Zeit* zu sortieren. Hier ist die Liste dieser Ergebniskombination:

lfd. Nr.	Fehler	Zeit
1	Ø	5Ø
2	Ø	56
3	Ø	53
4	Ø	62
5	Ø	49
6	Ø	55
7	Ø	57
8	Ø	6Ø
9	Ø	58
1Ø	Ø	48

Möglichkeit 2

Alle Teilnehmer haben *unterschiedliche Fehler* herausgeritten. In diesem Falle ist die Ergebnisliste nach Durchlaufen des vorigen Programms – das sie nach *Fehler* sortiert hatte – bereits das gewünschte Endergebnis Ihrer Bemühungen. Diese Liste würde also so aussehen:

lfd. Nr.	Fehler	Zeit
1	Ø	55
2	4	5Ø
3	8	55
4	12	6Ø
5	16	7Ø
6	2Ø	75
7	24	77
8	28	74
9	3Ø	77
1Ø	36	1ØØ

Möglichkeit 3

Es gibt Gruppen von fehlergleichen Reitern, die jedoch für ihre Vorstellung unterschiedliche Zeiten benötigt haben. Dieser Fall lag in der DATA-Liste auf Seite 202 unserem letzten Programm zugrunde. Ich zeige Ihnen unten noch einmal diesen Teil einer möglichen Computerausgabe (links); daneben steht, wie die Ergebnisliste nach den Regeln aussehen müßte.

Das hat der Computer ausgegeben:

lfd. Nr.	Fehler	Zeit
1	Ø	52
2	Ø	5Ø
3	Ø	6Ø
4	Ø	55
5	Ø	58
6	4	61
7	4	62
8	4	6Ø
9	8	63
1Ø	8	65

So muß die Liste nach den Regeln sein:

lfd. Nr.	Fehler	Zeit
1	Ø	5Ø
2	Ø	52
3	Ø	55
4	Ø	58
5	Ø	6Ø
6	4	6Ø
7	4	61
8	4	62
9	8	63
1Ø	8	65

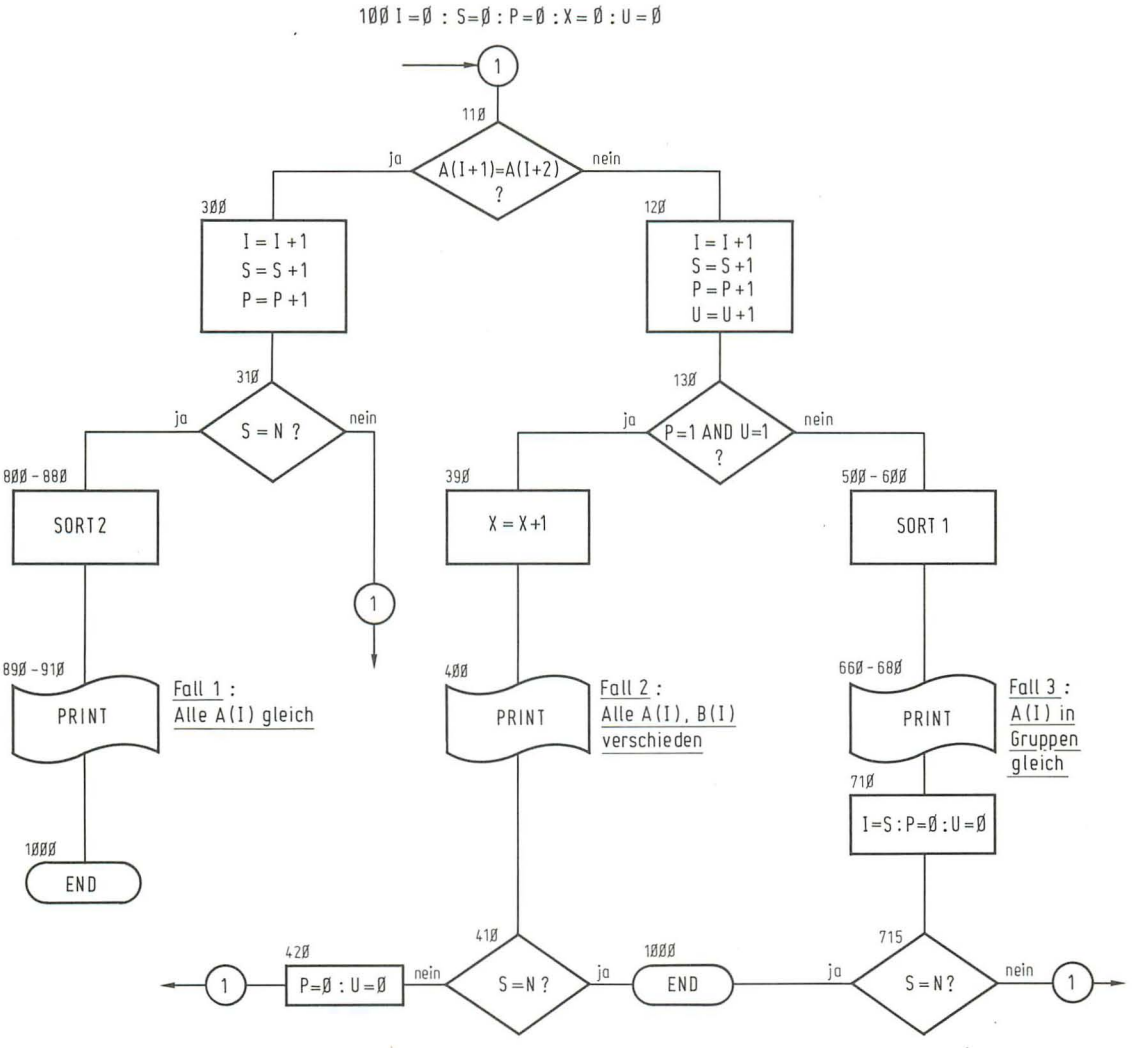


Abb. 43 Flußplan zur Ermittlung des Siegers

Es gilt nun, ein Programm zu entwickeln, das für alle drei möglichen Kombinationen immer die richtige Reihenfolge für die Bewertung des Springturniers ausdruckt.

In der folgenden *Abb. 43* zeige ich Ihnen zunächst den Ablaufplan. Zum besseren Verständnis sollen Sie noch einmal die Seiten 15...20 durchlesen; ferner sollten Sie wissen, daß jetzt die Fehler mit A(I), die Zeit mit B(I) verschlüsselt sind.

Das Programm dazu hat folgenden Wortlaut:

* Zeilen 5...5Ø Vorbereitung und Eingabeteil

```
 5 CLEAR 1ØØØ
1Ø CLS
15 DIM A(2Ø)
2Ø DIM B(2Ø)
25 N = 1Ø
3Ø FOR I = 1 TO N
35 INPUT "A(I) =";A(I)
4Ø INPUT "B(I) =";B(I)
45 NEXT I
5Ø PRINT "I =","A(I) =","B(I) ="
6Ø PRINT"==========================================
======"
```

* Durchsuchen der A(I) auf gleiche A(I)

```
1ØØ I = Ø : S = Ø : P = Ø : X = Ø : U = Ø
11Ø IF A(I+1) = A(I+2) THEN 3ØØ
12Ø I = I+1 : S = S+1 : P = P + 1 : U = U + 1
13Ø IF P = 1 AND U = 1 THEN 39Ø
14Ø GO TO 5ØØ
3ØØ I = I+1 : S = S+1 : P = P+1
310 IF S = N THEN 8ØØ
32Ø GO TO 11Ø
39Ø X = X+1
4ØØ PRINT I,A(I),B(I)
41Ø IF S = N THEN 1ØØØ
42Ø P = Ø : U = Ø
43Ø GO TO 11Ø
```

* Zeilen 5ØØ...6ØØ Sortierprogramm 1 für den Fall 3

```
5ØØ L = S − P+1
5Ø5 X = X+P
51Ø Z = P−1
52Ø FOR I = L TO X−1
53Ø FOR Y = 1 TO Z
54Ø IF B(I) < B(I+Y) THEN 58Ø
```

```
550 H = B(I) : H1 = A(I)
560 B(I) = B(I+Y) : A(I) = A(I+Y)
570 B(I+Y) = H : A(I+Y) = H1
580 NEXT Y
590 Z = Z - 1
600 NEXT I
```

* Zeilen 660...680 Ausgabe der Liste für den Fall 3; Rücksetzen der Variablen

```
660 FOR I = L TO X
670 PRINT I ,A(I), B(I)
680 NEXT I

710 I = S : P = 0 : U = 0
720 GO TO 110
```

* Zeilen 800...880 Sortierprogramm 2 für den Fall 1

```
800 Z = N - 1
810 FOR I = 1 TO N-1
820 FOR Y = 1 TO Z
830 IF B(I) < B(I+Y) THEN 865
840 H = B(I) : H1 = A(I)
850 B(I) = B(I+Y) : A(I) = A(I+Y)
860 B(I+Y) = H : A(I+Y) = H1
865 NEXT Y
870 Z = Z - 1
880 NEXT I
```

* Zeilen 890...910 Ausgabe der Liste für den Fall 1

```
 890 FOR I = 1 TO N
 900 PRINT I,A(I),B(I)
 910 NEXT I

1000 END
```

Sie sehen, daß oft kleine Probleme einen großen Lösungsaufwand erfordern! Betrachten Sie vor allem die Sortierprogramme noch einmal genau anhand der Abb. 43! Obwohl es sich um schon oft benutzte Standard-Programme handelt, unterscheiden sie sich doch stark in den Eingangszeilen 500...530 bzw. 800...820.

In diesen Zeilen wird jeweils festgelegt, in welchen Grenzen zu sortieren ist. Ich muß an dieser Stelle noch einmal meinen Standard-Ratschlag anbringen: Prüfen Sie immer genau, unter welchen Umständen Sie ein schon fertiges Programmstück aus einem alten in ein neues Programm übertragen können!

Haben Sie das Programm fehlerfrei übertragen, sollten Sie zur Kontrolle nacheinander die drei Listen (Fall 1...Fall 3) eingeben und sich davon überzeugen, daß das Programm für alle drei Fälle die richtigen Siegerlisten ausgibt!

● Sie sind immer noch nicht fertig!

Im Leben – in einem wirklichen Reitturnier – sind auch andere Ergebnisse denkbar, als wir sie in den Fällen 1...3 zugrundegelegt hatten.

So können – beispielsweise – durchaus zwei Reiter ihre Aufgabe fehler- *und* zeitgleich abgeschlossen haben. Sie sehen etwa in der folgenden Liste, daß die Reiter 2 und 3 jeder einen „0-Fehler-Ritt" in 50 Sekunden hingelegt haben.

In einem solchen Falle bestimmen die Regeln, daß es *zwei* zweite Plätze gibt; dafür keinen dritten. Sehen Sie sich die Liste an:

Das hat der Computer nach den vorigen Programmen ausgegeben:			So muß nach den Regeln die Siegerliste sein:			
lfd. Nr.	Fehler	Zeit	lfd Nr.	Rang	Fehler	Zeit
1	Ø	4Ø	1	1	Ø	4Ø
2	Ø	5Ø	2	2	Ø	5Ø
3	Ø	5Ø	3	2	Ø	5Ø
4	Ø	6Ø	4	4	Ø	6Ø
5	4	7Ø	5	5	4	7Ø
6	4	7Ø	6	5	4	7Ø
7	8	8Ø	7	7	8	8Ø
8	8	9Ø	8	8	8	9Ø
9	8	9Ø	9	8	8	9Ø
1Ø	12	1ØØ	1Ø	1Ø	12	1ØØ

Es gilt also, ein Programm zu entwickeln, das die Rangfolge so ausgibt, wie das oben eingerahmt ist! Auch dieses Problem wollen wir der besseren Übersichtlichkeit halber isoliert angehen.

* Zeilen 5...64: neuer Ein- und Ausgabeteil

```
 5 CLEAR 1ØØ
1Ø CLS
15 DIM A(2Ø)
2Ø DIM B(2Ø)
3Ø N = 1Ø
4Ø FOR I = 1 TO N
45 INPUT "A(I) =";A(I)
5Ø INPUT "B(I) =";B(I)
6Ø NEXT I
62 PRINT "LFD.-NR","RANG","FEHLER","ZEIT"
64 PRINT"==========================================
======"
```

* Zeilen 7Ø...5ØØ Ermittlung der Rangfolge

```
 7Ø I = 1 : N(I) = I
 75 Z = Ø
 8Ø PRINT I,N(I),A(I),B(I)
 85 IF I = N THEN 5ØØ
 9Ø I = I + 1
1ØØ IF A(I) = A(I − 1) AND B(I) = B(I − 1) THEN 15Ø
12Ø Z = Ø
13Ø N(I) = I
14Ø GO TO 8Ø
15Ø I = I
155 Z = Z + 1
16Ø N(I) = I − Z
17Ø GO TO 8Ø
5ØØ END
```

Das zugehörige Flußdiagramm finden Sie in *Abb. 44*.

Nun wäre es ja eine reizvolle Aufgabe für Sie, alle Teilprogramme zum Thema „Springturnier" zu einem einzigen zusammenzufassen. Aber das sollten Sie nur tun, wenn Sie wirklich per Computer ein Turnier abwickeln wollen; ansonsten sollten Sie sich mit dem Übungseffekt begnügen!

● Wir fassen zusammen!

Damit Sie vor lauter Reitern und Rössern Ihr Ziel nicht aus den Augen verlieren, wollen wir einmal zusammenfassen, was Sie am Beispiel der Reiterei zum Thema „Datenverarbeitung" eingeübt haben:

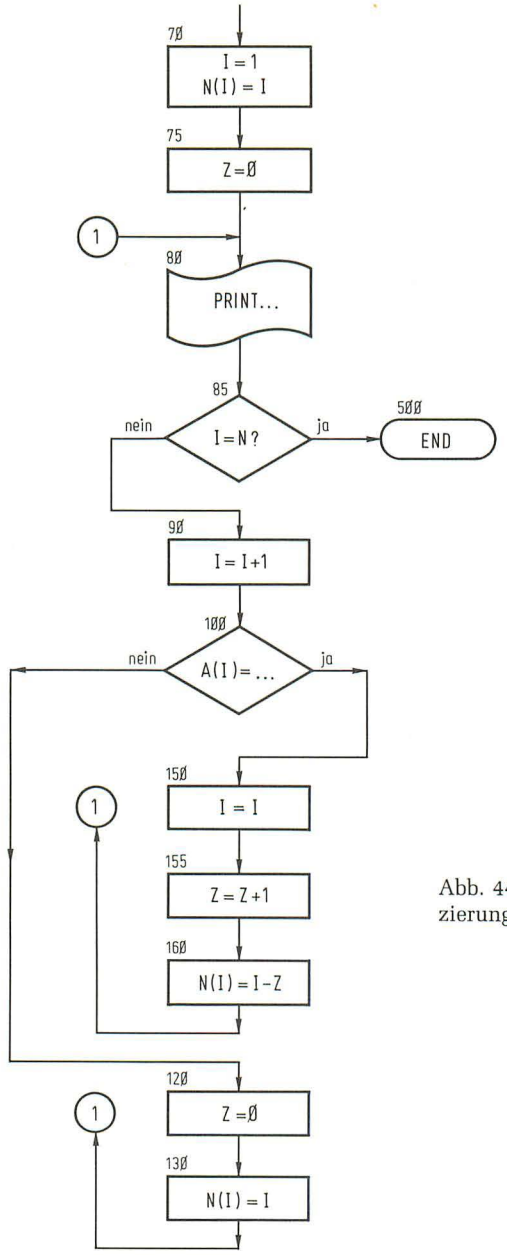


Abb. 44 Flußplan für das „Plazierungsprogramm“

– Sie haben eine Liste von Namen (Daten!) in alphabetischer Reihenfolge sortiert. Vor dem Ausdruck der sortierten Liste haben Sie jedem Namen eine bestimmte Nummer zugeordnet. Diese hat eine andere Bedeutung als bloß eine „laufende

Nummer"; war die Nummer erst generiert, gehörte sie von da an zu dem Namen wie etwa Ihre Paßnummer zu Ihrem (und nur zu Ihrem!) Paß.

- Sie haben die gleiche Liste ein zweites Mal nach einem anderen Kriterium sortiert. Dazu haben Sie – nach einigen Manipulationen – das gleiche Sortierprogramm noch einmal verwendet. Zusätzlich haben Sie die Liste auf mehrfach vorkommende Namen bzw. Begriffe durchsucht.
- Sie haben eine Liste von Namen (Daten!) nach einem bestimmten Merkmal (im Beispiel: Teilnahme an einer bestimmten Prüfung) durchsucht. Sie haben sicher Phantasie genug, sich vorzustellen, daß man eine Liste auch nach anderen Begriffen durchsuchen kann:
  ...Wer ist katholisch?
  ...Wer ist Ingenieur?
  ...Wer hat seinen Clubbeitrag noch nicht bezahlt? usw.
  Sie müssen nur den einzelnen Merkmalen oder Eigenschaften bestimmte Kennzahlen oder -buchstaben zuordnen und suchen dann nach diesen! Bei der Ausgabe dieser Liste haben Sie aus den erklärten Gründen nicht mit „A", sondern einem anderen, aufgabenspezifischen Buchstaben begonnen.
- Im Rahmen des Springturniers hatten Sie die anfallenden Daten Ihrem Computer über eine DATA-Liste anvertraut. Sie hatten die Daten sortiert und einige – im Beispiel die ausgeschiedenen Reiter – vor der Ausgabe ausgesondert. Auch dazu hatten Sie im Programm ebenso wie in der Datenliste mit Kennzahlen (999) gearbeitet.
- Eine bereits nach einem Merkmal (Fehlerzahl) sortierte Liste hatten Sie ein zweites Mal nach einem anderen Merkmal (Zeit) sortiert.
- Schließlich hatten Sie nach den vorgegebenen Regeln die sortierten Daten in eine Rangfolge gebracht.

Es ist wahrscheinlich, daß Sie alle Detailaufgaben bei nur zehn oder 15 Teilnehmern schneller „zu Fuß" erledigt hätten. Ein zeitaufwendiges Computerprogramm lohnt sich also nur bei vielen Teilnehmern (Daten!) oder nur dann, wenn das Programm öfter genutzt werden soll.

Bleibt noch etwas anzumerken?

Ich denke, Sie haben genug Vorstellungskraft, die Grundgedanken der Übungsprogramme auf ganz andere Aufgaben zu übertragen. Nutzen Sie dabei Programme oder Programmteile, die Sie schon in Ihrer Programmbibliothek haben! Aber tun Sie es mit Verstand! Für umfangreichere Datensätze erinnern Sie sich an CLEAR.. und DIM.. und setzen Sie dies nach den Regeln ein!

Schließlich: Bei an der Mathematik orientierten Programmen haben Sie gelernt, daß BASIC *allein* nicht glücklich macht; Sie müssen die Formeln und Zusammenhänge kennen. Hier hätten Sie mit noch soviel BASIC-Wissen kein brauchbares Programm zustandegebracht, wenn Ihnen die Spielregeln der Reiterei nicht geläufig gewesen wären!

14.5 Matrizen-Operationen

Was ist eine Matrix? Das ist eine Matrix:

1	2	3	8
4	9	12	14
6	4	$\boxed{5}$	6
7	7	9	1Ø
1	23	12	14

Die Zahlen in der obigen Anordnung sind belanglos. Für 1 können Sie ebenso gut .325 oder 1234 schreiben.

Wichtig für Ihre weitere Arbeit ist:

- Eine Matrix ist eine Anordnung von "Daten", die zeilen- und spaltenweise geordnet sind. Die Matrix im Beispiel oben hat
  5 Zeilen und
  4 Spalten.
- Alle Elemente einer Matrix haben den gleichen Namen (beispielsweise A).
- Der Platz jeden Elementes innerhalb der Matrix wird durch Zeilen- und Spalten-*Indizes* bestimmt. Also beispielsweise A(Z,S).
- Beachten Sie bei der Indizierung, daß immer
  zuerst die *Zeile*
  danach die *Spalte*
  genannt werden.
  Die Zahl 5 in der obigen Matrix steht beispielsweise in der *dritten Zeile* und in der *dritten Spalte*; demzufolge müßte es heißen: A(3,3) = 5.

Mit dem folgenden Programmbeispiel sind Sie zunächst in der Lage, in Ihren Computer Matrizen beliebiger Zeilen- und Spaltenzahl zu laden.

```
1Ø CLEAR 1ØØØ : CLS
2Ø INPUT"WIEVIELE ZEILEN SOLL DIE MATRIX HABEN ?";Z
3Ø INPUT"WIEVIELE SPALTEN SOLL DIE MATRIX HABEN ?";S
35 DIM A(Z, S) : CLS
```

Nach dieser „Matrix-Definition" erfolgt das Einlesen der Elemente A(I,J) wie folgt:

```
4Ø FOR I = 1 TO Z
5Ø PRINT"FUER DIE ZEILE :";I
6Ø PRINT"==========================================
====="
```

```
7Ø FOR J = 1 TO S
8Ø PRINT"BETRAEGT DER WERT FUER DIE SPALTE :";J
9Ø INPUT A(I,J)
1ØØ NEXT J
1Ø4 CLS
1Ø8 NEXT I
11Ø PRINT"************ENDE DER EINGABE !************"
```

Bevor Sie die Matrix „laden", schauen Sie sich noch den folgenden Programmteil für die Ausgabe an! Dieser Teil hängt nämlich davon ab, wie Sie in Zeile 3Ø die Frage nach der *Spaltenzahl* beantwortet hatten.

Sie haben in Zeile 3Ø für S eingegeben:	Das Ausgabeprogramm lautet dann:
2	2ØØ FOR I = 1 TO Z 21Ø PRINT A(I,1),A(I,2) 23Ø NEXT I
3	2ØØ FOR I = 1 TO Z 21Ø PRINT A(I,1),A(I,2),A(I,3) 23Ø NEXT I
4	2ØØ FOR I = TO Z 21Ø PRINT A(I,1),A(I,2),A(I,3),A(I,4) 23Ø NEXT I Nur für eine 4-Spaltenmatrix geht auch die folgende Sequenz; wobei es auf das Komma am Ende der Zeile 21Ø ankommt: 2ØØ FOR I = 1 TO Z 2Ø5 FOR J = 1 TO S 21Ø PRINT A(I,J), 22Ø NEXT J,I
5	2ØØ FOR I = 1 TO Z 21Ø PRINT TAB(Ø)A(I,1);TAB(5)A(I,2);TAB(1Ø)A (I,3);TAB(15)A(I,4);TAB(2Ø)A(I,5) 23Ø NEXT I
N*) (eine Zahl größer als 5!)	2ØØ FOR I = 1 TO Z 21Ø PRINT TAB(Ø)A(I,1) . . . . . TAB(...)A(I,N)

Ich glaube, Sie sehen genau, daß das „Ausgabeformat" nur in der Zeile 21Ø festgelegt wird; dieses Ausgabeformat muß mit Ihrer Eingabe übereinstimmen.

*) Überfordern Sie Ihren Computer nicht! Er faßt nur 64 Zeichen/Zeile. Wollen Sie in jeder Spalte acht Zeichen inkl. Zwischenraum unterbringen, dann darf N max. 8 werden!

Berühmte Ausnahme von der Regel: Sie wollen ausdrücklich etwas anderes!

Um für unsere ersten Fingerübungen etwas Zeit zwischen Ein- und Ausgabe zu schaffen, fügen Sie noch ein:

```
15Ø FOR X = 1 TO 1ØØØ NEXT : CLS
```

Und nun RUNnen Sie los! Beantworten Sie die Frage des Computers nach der Zeilenzahl mit 5, die nach der Spaltenzahl mit 4 und geben Sie unsere Versuchsmatrix zeilen- und spaltenweise ein! Immer hübsch den Befehlen des Computers gehorchen! Nach der durch Zeile 15Ø erzeugten „Kunstpause" sollten Sie die Matrix so auf Ihrem Bildschirm sehen, wie sie oben steht!

Im nächsten Schritt wollen wir untersuchen, ob man auch mit *Strings* Matrizen aufbauen kann. Ändern Sie dazu nur die Zeilen:

```
35 DIM A$(Z, S) : CLS
9Ø INPUT A$(I,J)
21Ø PRINT TAB(Ø)A$(I,1);TAB(5)A$(I,2);TAB(1Ø)A$(I,3);TAB(15)A$(I,4);
    TAB(2Ø)A$(I,5)
```

Für Zeilen- und Spaltenzahl geben Sie je 5 ein; ferner die folgende Matrix, die Sie dann auch genau so auf Ihrem Schirm sehen werden.

A	B	C	D	E
F	G	H	I	K
L	M	N	O	P
Q	R	S	T	U
V	W	X	Y	Z

Es versteht sich, das Sie für A auch ANTON und für Z auch ZEPPELIN eingeben können. (Falls Sie gern Schreibmaschine schreiben!)

Und nun prüfen wir noch, ob in einer Matrix auch Strings und numerische Variable gemischt vorkommen dürfen!

Wir wollen in einer 4×3-Matrix in der ersten und dritten Spalte einen String, in der zweiten eine Zahl unterbringen. Dazu ändern wir das Programm ab.

```
1Ø CLEAR 1ØØØ : CLS
2Ø Z = 4 : S = 3
3Ø FOR I = 1 TO Z
4Ø PRINT"NAME IN ZEILE";I;" ,SPALTE 1"
45 INPUT N$(I,1)
5Ø PRINT"ZAHL IN ZEILE";I;" ,SPALTE 2"
55 INPUT N(I,2)
6Ø PRINT"NAME IN ZEILE";I;" ,SPALTE 3"
65 INPUT N1$(I,3)
67 CLS
```

```
70 NEXT I
100 FOR X = 1 TO 1000 : NEXT : CLS
200 FOR I = 1 TO Z
210 PRINT N$(I,1),N(I,2),N1$(I,3)
220 NEXT I
```

Starten Sie das Programm und beantworten Sie die Fragen des Computers wie folgt:

BACH	2	MEIER
MEIER	2	BACH
BAUM	7	APFEL
BIRNE	12.5	OBST

Ich denke, die bisherigen Beispiele führen Ihnen deutlich vor Augen, wie man Matrizen in den Computer hinein- und wieder herausschaufelt!

14.5.1 Wir arbeiten mit Matrizen

Natürlich hat das bloße Ein- und Auslesen allein keinen rechten Sinn; wir kommen nun dazu, mit der Matrizentechnik etwas sinnvolles zu tun.

```
5 CLS : Z = 4 : S = 4
20 FOR X = 1 TO Z
30 FOR Y = 1 TO S
40 INPUT A(X,Y)
50 NEXT Y,X
```

Die Zeilen 5...50 stellen den bereits bekannten Eingabeteil dar. Geben Sie 4 × 4 Zahlen ein; wählen Sie solche zwischen 1 und 10, damit Sie die folgenden Übungen besser verfolgen können.

```
100 CLS
110 FOR X = 1 TO Z
120 FOR Y = 1 TO S
130 PRINT A(X,Y),
140 NEXT Y,X
```

Mit 100...140 erhalten Sie bekanntlich die Matrix in der gleichen Form bzw. mit den gleichen Werten ausgegeben, wie Sie sie eingegeben hatten. Und nun ersetzen Sie nacheinander die Zeile 130 durch die folgenden:

```
130 PRINT A(X,Y) * 10,
130 PRINT A(X,Y)/2.5,
130 PRINT SQR(A(X,Y)),
130 PRINT INT(SQR(A(X,Y))),
```

Ich glaube, Sie sehen deutlich wie das funktioniert! Wissen Sie noch, warum die Zeile 13Ø *hier* mit einem Komma abgeschlossen werden muß? (Lassen Sie's doch mal weg!)

Für die nächste Übung wollen wir einen Aufgabentext formulieren:

- Es soll eine 5×3-Matrix eingegeben werden.
- Es soll eine 5×4-Matrix ausgegeben werden.
- Dabei sollen die Werte in der 4. Spalte die Produkte der Werte in Spalte 2 und 3 sein.
- Schließlich soll am Schluß in einer Extra-Zeile die Summe aller Werte der Spalte 4 ausgeschrieben werden.

Das ist viel, viel einfacher, als Sie befürchten!

Wir gehen von dem letzten Programm aus und ändern dort nur:

```
5 CLS : Z = 5 : S = 3
```

Die oben formulierten Aufgaben erledigen wir im *Ausgabe*-Teil so:

```
1Ø5 S = Ø
12Ø A(X,4)=A(X,2)*A(X,3)
13Ø S = S + A(X,4)
14Ø PRINT A(X,1),A(X,2),A(X,3),A(X,4)
15Ø NEXT X
16Ø PRINT:PRINT"DIE SUMME DER A(X,4) IST :";S
```

Wie Sie bemerken werden, wird die Summe in den Zeilen 1Ø5 und 13Ø, das Produkt in Zeile 12Ø gebildet. Alles klar?

Mit dem folgenden Programm wollen wir die anspruchsvolle Aufgabe des Sortierens – diesmal innerhalb von Matrizen – einüben.

```
* 1Ø...14Ø enthalten nach den schon bekannten Mustern den Eingabeteil.
1Ø CLS
3Ø INPUT"ZEILEN";Z
4Ø INPUT"SPALTEN";S
5Ø CLS
6Ø FOR I = 1 TO Z
7Ø PRINT"FUER DIE ZEILE :";I
75 FOR J = 1 TO S
8Ø PRINT"WERT FUER DIE SPALTE :";J
9Ø INPUT A(I,J)
```

```
110 NEXT J
120 CLS
130 NEXT I
140 PRINT"ENDE DER EINGABE"
```

* Zeilen 200...270: Deren Geheimnis erfahren Sie später!

```
200 S = 1
210 GOSUB 1000
220 S = 2
230 GOSUB 1000
240 S = 3
250 GOSUB 1000
260 S = 4
270 GOSUB 1000
```

* Die Zeilen 495...530 besorgen die Ausgabe!

```
495 CLS
500 FOR I = 1 TO Z
510 PRINT A(I,1),A(I,2),A(I,3),A(I,4)
520 NEXT I
530 END
```

* In den Zeilen 1000...1100 wird sortiert! Beachten Sie die Zeilen 1030 und 1040 ...1060 und vergleichen Sie diese mit den korrespondierenden Zeilen längst bekannter Sortierprogramme! Da kann man nur sagen: „andere Techniken, andere ‚Sitten'"!

```
1000 R = Z - 1
1010 FOR W = 1 TO Z - 1
1020 FOR Q = 1 TO R
1030 IF A(W,S) < A(W+Q,S) THEN 1070
1040 H = A(W,S)
1050 A(W,S)=A(W+Q,S)
1060 A(W+Q,S)=H
1070 NEXT Q
1080 R = R - 1
1090 NEXT W
1100 RETURN
```

Wenn Sie sich von den Ergebnissen des Programms verblüffen lassen wollen, empfehle ich Ihnen, folgende Matrix einzugeben:

1. Z = 1Ø; S = 4

1Ø	2	3	8
2	1	7	6
4	3	1Ø	3
7	5	6	1Ø
1	7	8	7
5	1Ø	2	2
8	9	5	1
3	8	9	9
6	4	4	4
9	6	1	5

Na, sind Sie verblüfft?

Sie müssen auf Ihrem Schirm vier Zahlenkolonnen sehen, die alle hübsch von 1...1Ø sortiert sind!

Verantwortlich für dieses Ergebnis sind die Programmzeilen 2ØØ...27Ø, nach denen viermal hintereinander jeweils die Spalten S = 1...S = 4 durch das Sortierprogramm gejagt werden!

Merken Sie sich zu dem letzten Programm vor allem im Hinblick auf eine spätere Verwendung in anderem Zusammenhang:

- Statt aller vier Spalten können Sie natürlich nur eine oder zwei sortieren. Sie bestimmen das in den Zeilen 2ØØ...27Ø.
- Beachten Sie, daß in diesem Programm beim Sortieren keine Vertauschungen der Spaltenwerte einer Zeile vorgenommen werden. Würden Sie also nur die Werte der Spalte 1 durch das Sortierprogramm laufen lassen, dann würden die Werte der Spalten 2...4 so ausgegeben, wie sie eingegeben waren.
- Soll das nicht so sein – wollen Sie also etwa Spalte 1 zwar sortieren, die Werte der anderen Spalten auf der gleichen Zeile aber „umstellen", dann müssen Sie die „Vertauschungszeilen" 1Ø4Ø...1Ø6Ø nach dem Muster früherer Sortierprogramme entsprechend gestalten!
- Die hier praktizierte „Mehrfachnutzung" eines Sortierprogrammes funktioniert nur, wenn alle Elemente der Matrix vom gleichen Typ sind. Also entweder alle A(...) oder alle A$(...). Liegt eine gemischte Matrix vor, dann können Sie das Programm trotzdem nutzen! Sie müssen einfach numerische Variable als Stringvariable eingeben und bearbeiten. Also würde im Beispiel aus A(...) der String A$(...).

14.5.2 Sie handeln mit Obst und Matrizen

In der nächsten Übung spielen Sie einen Obsthändler, der mit Hilfe der Matrizentechnik seinen Tagesumsatz ermitteln will. Er hat am Abend in seinen Computer folgende Liste eingegeben:

ARTIKEL	VERKAUFTE STUECK	EINZELPREIS
Zwetschgen	7	1.6
Birnen	2	3.2
Orangen	10	10.9
Äpfel	8	9.3
Zitronen	10	17.1

Er verlangt von seinem Computer:
- Die Liste ist nach Artikelnamen zu sortieren
- Es ist in einer 4. Spalte der Verkaufserlös jeder Artikelart zu errechnen und auszugeben.
- Am Ende der Liste ist der Gesamtumsatz auszuwerfen.

Packen Sie's an!

```
5 CLEAR 1000 : DIM A$(15,4) : DIM N(15,4) : DIM N1(15,4) : DIM E(15,4)
10 N = 5
15 FOR I = 1 TO N
17 CLS
20 INPUT"ARTIKEL=";A$(I,1)
30 INPUT"VERKAUFTE STUECK =";N(I,2)
40 INPUT"EINZELPREIS =";N1(I,3)
50 NEXT I

100 CLS: PRINT"ARTIKEL", "STUECK","E.-PREIS","ERLOES"
110 PRINT"=====================================
======"
120 W = 0
130 FOR I = 1 TO N -1
140 IF A$(I,1) < = A$(I+1,1) THEN 210
150 W = 1
160 FOR J = 1 TO 3
170 H$=A$(I,J):H=N(I,J):H1=N1(I,J)
180 A$(I,J)=A$(I+1,J):N(I,J)=N(I+1,J):N1(I,J)=N1(I+1,J)
190 A$(I+1,J)=H$:N(I+1,J)=H:N1(I+1,J)=H1
200 NEXT J
```

```
210 NEXT I
220 IF W = 1 THEN 120
300 S = 0
310 FOR I = 1 TO N
320 E(I,4)=N(I,2)*N(I,3)
330 S = S + E(I,4)
340 PRINT A$(I,1),N(I,2),N1(I,3),E(I,4)
350 NEXT I
360 PRINT:PRINT
370 PRINT"GESAMT-UMSATZ =";S;" DM"
380 PRINT"=====================================
```

Mit Ausnahme des Sortierprogramms – das ich Ihnen in einer Variante des schon bekannten untergeschmuggelt habe; BASIC ist ja sooo vielseitig! – können Sie das Programm wohl selbst nachvollziehen, da es alle Varianten verwendet, die Sie soeben gelernt haben. Und für das Sortierprogramm hatte ich mir gedacht, daß Sie es selbst einmal aus den bekannten (etwa Seite 137) ableiten. Keine Lust?

14.5.3. Und nun sind Sie Malermeister

Sie verkaufen in Ihrem Laden Farben. Im Tagesgeschäft haben Sie keine Zeit, diejenigen Farbbüchsen, die Sie den Kunden zwar gezeigt, die diese aber nicht abgenommen haben, wieder ins Lager einzuräumen; Sie stellen sie einfach auf einen Tisch und sortieren sie am Wochenende mit Hilfe Ihres Computers ein.

Sie haben zwei Arten von Farben:	Grundierung	Kennziffer 1
	Decklack	Kennziffer 2
Ihre Farben verkaufen Sie in drei Gebindegrößen:	0,5-Liter-Dosen	Kennziffer 1
	1-Liter-Dosen	Kennziffer 2
	2,5-Liter-Dosen	Kennziffer 3
Und schließlich verfügen Sie über eine Palette von vier Farben:	weiß	Kennziffer 1
	rot	Kennziffer 2
	grün	Kennziffer 3
	blau	Kennziffer 4

In Ihrem Ordnungssystem hat also der Decklack in der 1-Liter-Dose und in der Farbe weiß den Schlüssel: 2 – 2 – 1. Ist das klar?

Die Bedienung Ihres Computers besteht also darin, jede Büchse in die Hand zu nehmen, den Schlüssel zu lesen und wie folgt in den Computer zu übertragen: (Tun Sie's meinetwegen gleich!)

```
1000 DATA 2,3,3
1010 DATA 2,3,2
1020 DATA 1,2,1
1030 DATA 2,1,4
1040 DATA 1,2,1
1050 DATA 1,2,1
1060 DATA 2,3,2
1070 DATA 1,1,1
1080 DATA 2,1,4
1090 DATA 1,2,1
```

Wie Sie das gelernt haben, teilen Sie Ihrem Computer das Ende der Liste mit durch:

```
1100 DATA 9999,9999,9999
```

Und nun zum Programm! Zunächst lassen Sie nach altem Brauch Ihren Computer die Farbbüchsen zählen!

```
10 N = 0
20 READ A
30 N = N + 1
40 IF A = 9999 THEN 50
45 GO TO 20
```

Ich darf Sie daran erinnern, daß wir uns noch im Kapitel „Matrix-Operationen" befinden. Es gilt jetzt, die Matrix zu dimensionieren. Sie überblicken sofort, daß Sie insgesamt zehn verschiedene Farbbüchsen eingeräumt haben; Ihre Matrix also zehn Zeilen haben muß. Und da Ihre Farbbüchsen die drei Unterscheidungsmerkmale Art, Gebindegröße und Farbe haben, werden wohl drei Spalten herauskommen. Sie müssen Ihrem Computer helfen, das auch herauszufinden!

```
50 N = N - 1
```

In Zeile 50 ziehen Sie die 9999er Zeile ab, damit Ihr Apparat das nicht für eine noch unbekannte Farbsorte hält.

Die Zahl der Spalten können Sie Ihrem Computer ruhig im Klartext verraten, da sich diese ja nie ändern kann; auch wenn Sie 100 Büchsen einräumen. Tun Sie das mit:

```
52 S = 3
```

Die Zeilenzahl der Matrix muß nun Herr Computer selbst berechnen! (Denken Sie daran: das Programm soll allgemeingültig werden; also auch für 123 Büchsen gelten!)

Er dividiert einfach die Anzahl der in Zeile 20 gelesenen Daten (das waren 30) durch die Anzahl der von Ihnen in Zeile 52 vorgegebenen Spalten. Und zwar so:

```
54 Z = N/S
```

Und nun kann „Er" ganz einfach seine Matrix dimensionieren!

```
100 DIM A(Z,S)
```

Machen Sie einen Probelauf, damit Sie mir glauben! Und zwar mit:

```
150 RESTORE
160 FOR I = 1 TO Z
170 READ A (I,1),A(I,2),A(I,3)
180 PRINT A(I,1),A(I,2),A(I,3)
190 NEXT I
```

Zufrieden?

* In den Zeilen 500...610 folgt jetzt das Sortierprogramm.

```
500 W = 0
510 FOR I = 1 TO Z -1
520 IF A(I,S) < = A(I+1,S) THEN 590
530 W = 1
540 FOR Y = 1 TO 3
550 H=A(I,Y)
560 A(I,Y)=A(I+1,Y)
570 A(I+1,Y)=H
580 NEXT Y
590 NEXT I
600 IF W = 1 THEN 500

610 RETURN
```

Für die endgültige Ausgabe überschreiben Sie Ihr „Kontrollprogramm":

```
150 RESTORE
160 FOR I = 1 TO Z
170 READ A(I,1)A(I,2)A(I,3)
180 NEXT I
190 CLS
200 PRINT"ART","GEBINDE","FARBE"
210 PRINT"=====================================
======"
215 S = 1
220 GOSUB 500
230 FOR I = 1 TO Z
240 PRINT A(I,1),A(I,2)A(I,3)
250 NEXT I
```

Das Ergebnis dieses Programms wird eine Liste sein, die nach Farb-*Arten* sortiert ist.

Nun sind Sie aber etwas anspruchsvoller geworden, Sie wollen die Liste nach Farb-Arten sortiert haben; jedoch innerhalb jeder Farb-*Art* soll (zusätzlich) nach *Gebindegrößen* sortiert sein!

Dazu fügen Sie schlicht und einfach in Ihr Programm ein:

```
212 S = 2
214 GOSUB 500
```

Ist das nicht umwerfend (einfach)?

Erlauben Sie mir, Sie zum Abschluß noch einmal gehörig zu verblüffen?

Dann ändern Sie doch das vorige Programm einmal in folgenden Zeilen durch schlichtes Überschreiben ab:

```
 52 S = 2
170 READ A(I,1),A(I,2)
200 PRINT"FEHLER","ZEIT"
240 PRINT A(I,1),A(I,2)
540 FOR Y = 1 TO 2
1000 DATA 8,65,0,52,8,63,0,50,0,60,4,62,4,61,12,67,0,58,0,55,4,60,9999,9999
```

Bitte löschen Sie die DATA-Zeilen 1010...1100!

Nun starten Sie das Programm...
Betrachten es...
Schlagen Seite 206 auf...
Betrachten die dort stehende Liste...
Und dürften dann wohl sehr verblüfft sein!

Wie heißt es doch so treffend: Für zehn Probleme gibt es (in BASIC) mindestens elf Lösungen!

Ich wünsche Ihnen auch künftig viel Spaß mit BASIC!

Literaturverzeichnis

Level II Basic Reference Manual. Tandy Corp. Fort Worth, USA
Schärf, J.: Basic für Anfänger. R. Oldenbourg Verlag, München
Schärf, J.: Kunesch, A.: Basic für Kaufleute. R. Oldenbourg Verlag, München
Wittig, S.: Basic-Brevier. Verlag Heinz Heise, Hannover
Lien, D. A.: The Basic Handbook. Compusoft Publishing, San Diego, USA
Tracton, K.: 57 Practical Programs & Games in Basic. TAB Books, Blue Ridge Summit Pb. 17214, USA
Friedrich: Tabellenbuch für Elektrotechnik. Ferd. Dümmler Verlag, Bonn
Don Inman: TRS-80 Graphics. Tandy Corp. Fort Worth, USA
David A. Kater, Susan J. Thomas: TRS 80 Graphics. Tandy Corp. Fort Worth, USA
Lon Poole, Mary Borchers: 77 Basic-Programme. te-wi Verlag GmbH, München

Sachverzeichnis

A

Ablauf|fehler 122
–plan 23
Abschreibung 160
ABS (A) 35
Absolutbetrag 35
Achsenkreuz 90
Argument 12
ASC(...) 45
ASCII-Code 43, 44

B

Balkendiagramm 94
Befehl 11
Benutzerfunktion 31
Binärzahl 60
Bogenmaß 29

C

CHR$(...) 45
Codewort 59
Codieren 49
Concatenation 70
Cosinus 28, 29
Cotangens 30
Cursor 78

D

Daten|feld 74
–liste 68
Decodieren 49, 50
Declarationcharacter 37
Dimensionierung 144
DIN 66001 20
Durchschnittswert 132

E

EDIT 76
Editieren 76
Exponent 27
Exponentialfunktion 31

F

Falscheingabe 130
FIX(A) 35
Funktion 12

G

GET 54

H

Hochrechnung 168

I

INKEY$ 54
INT(A) 35
Integer 35

K

Kapital 157
Kommando 11
Kommentar 23
Koordinaten 93

L

LEFT$(STRING,N) 47
LEN(...) 46
Logarithmen 32
Logische Fehler 123

M

Matrix 214
Mehrfach-Nennungen 190
Menütechnik 129
Meßwerte 152
MID$(STRING;N,M) 48

O

Operator 13

P

POINT 107
Potenzschreibweise 26
PRINT ATX;A
– TAB(X) A 39
– USING 40
Programmablaufpläne 20
Programm-Listing 23
Pythagoras 148

R

RESET 87
RIGHT$(STRING,N) 48
Rundung 133

S

SET 87
SGN(A) 35
Sinus 28, 29
Sortieren 138
Spalte 214
Sperre 130
Statement 11
STR$(...) 53
STRING$(N,...) 52
Summe 132
Syntax-Fehler 122

T

Tangens 29
Teilnehmerliste 187
Tendenz 152
TROFF 124
TRON 124

U

Umwandlung 173

V

VAL(STRING) 52
Variable 37
Vereinzeln 133

W

Wertminderung 158
Winkelfunktionen 28
Wissenschaftliche Schreibweise 26

Z

Zählschleife 126
Zeile 214
Zinsen 157
Zufallszahlen 135

Weitere Franzis-Computer-Fachbücher

Basic-Interpreter

Funktionsweise und Implementierung in 8080/Z-80-Computern. Von Rolf-Dieter **Klein.**
2., verbesserte Auflage, 178 Seiten, 45 Abbildungen.
Lwstr-geb. DM 38.–
ISBN 3-7723-6942-1

Wie man 8080- oder Z-80-Systeme nachträglich mit einem Basic-Interpreter ausrüsten kann, beschreibt dieses Buch. Dabei werden mehrere Ausführungen erörtert und beschrieben. Die beiden interessantesten sind: Ein Tiny-Basic-Interpreter und ein komfortabler 12-KByte-Basic-Interpreter. Die Krönung bildet die ausführliche Beschreibung eines Basic-Interpreters für den 16-Bit-Prozessor Z8000.

Mikrocomputer selbstgebaut und programmiert

Vom Bauelement zum fertigen Z-80-Computer. Von Rolf-Dieter **Klein.**
316 Seiten mit 332 Abbildungen.
Lwstr-geb. DM 38.–
ISBN 3-7723-7161-2

Einen Mikrocomputer selbst zu bauen und zu programmieren, das ist das Ziel dieses Buches. Und das Ziel wird erreicht. Wie? Durch ein neuartiges Experimentiersystem. Die Platinen dazu sind im Handel erhältlich oder können selbst hergestellt werden. Durch ein modulares Konzept ergeben sich zahlreiche Kombinationsmöglichkeiten. Der Softwareteil des Bandes bietet für alle im Rechner verwendete Programme ausführliche Ausdrucke.

Was ist Pascal?

Eine einfache und kompakte Darstellung der Programmiersprache mit vielen Beispielen. Von Rolf-Dieter **Klein.**
124 Seiten, 72 Abbildungen.
Lwstr-geb. DM 32.–
ISBN 3-7723-7001-2

Dies ist eine praxisnahe Arbeitsanleitung von Anfang an mit Pascal zu programmieren. Von anderen Pascal-Büchern unterscheidet sich dieses dadurch, daß es auf die Erfordernisse der Mikrocomputer und Heimcomputer besonders eingeht.

Mikrocomputer Hard- und Softwarepraxis

Anhand ausführlicher Beispiele und größerer Programme wird das Programmieren immer perfekter. Von Rolf-Dieter **Klein.**
220 Seiten, 125 Abbildungen, 6 Tabellen.
Lwstr-geb. DM 38.–
ISBN 3-7723-6812-3

Hier geht es um den Z80, das Arbeitspferd unter den Mikrocomputern. Ihn hardwaremäßig gut zu verstehen, ihn softwaremäßig voll auszufahren, das bewirkt dieses Buch.
Der Band besteht aus drei Teilen, die deutlich miteinander verknüpft sind. Erstens, dem Hardware-Teil. Darin bespricht der Autor die zahlreichen Peripheriegeräte, die einen Z80 erst zum vollwertigen Mikrocomputer machen.
Zweitens, dem Softwareteil. Hier wird die wichtige System-Software so intensiv besprochen, daß sie dem Anwender möglichst schnell geläufig wird.
Drittens, dem Programmteil. Der lehrt das freie Programmieren, wiederum mit ausführlichen Programmlisten. Diese vermitteln dem Benutzer einen großen Erfahrungsschatz, den andere eingesammelt haben.

Software-Engineering und ihre Qualitätssicherung

Methoden zu erfolgreichen Problemlösungen für den Personalcomputer-Anwender. Von Friedrich **Haugg.**
141 Seiten mit 49 Abbildungen.
Lwstr-geb. DM 28.–
ISBN 3-7723-7181-7

Ab sofort kann der Personalcomputer-Anwender Software-Engineering und Software-Qualitätssicherung für seine vielfältigen Aufgaben nutzbar machen. Hier findet der Anwender, was er braucht: Methoden und Verfahren, die ihm helfen, gute Software termin- und kostengerecht herzustellen.
Der Autor geht von bekannten Methoden der großen Anwender aus. Was sie sich erfolgreich zunutze machen, das hat er hier auf die tatsächlichen Möglichkeiten der kleinen und mittleren Anwender übertragen.

Erfolgreicher mit CBM arbeiten

Für alle CBM-Anwender eine verständliche Einführung in die Maschinensprache. Von Dipl.-Ing. Franz **Wunderlich.**
148 Seiten, 8 Abbildungen.
Lwstr-geb. DM 36.–
ISBN 3-7723-7051-9

CBM-Anwender mit Basic-Erfahrung holen mit diesem Buch mehr aus ihrem Computer heraus. Sie können nämlich mit dem Programmieren in der Maschinensprache beginnen. Zunächst wird ihnen beigebracht, wie ein 6502 programmiert wird. Zahlensysteme, Speicherkonzepte, Adressierung und Befehlssätze werden behandelt und gewinnen Klarheit. Im Hauptteil wird speziell die geräteabhängige Software der Commodore-Serie CBM abgehandelt.
Natürlich bringt der Autor auch fertige Programme und viele CBM-spezielle Anwenderbeispiele im Anhang.

Mit Computer steuern

Aufbau und Anwendung von Einplatinen-Mikrocomputern. Von Herwig **Feichtinger.** 184 Seiten mit 144 Abbildungen. Lwstr-geb. DM 38.–
ISBN 3-7723-7221-X

Mit Einplatinencomputern steuern und regeln, ab wann sich das lohnt und wie das zu machen ist, darum geht es in diesem Buch. Der Autor zeigt, daß der technische Aufwand dafür gering ist.
Den Schwerpunkt legte der Autor auf die Beschreibung der Bauelemente und Baugruppen, also die Mikroprozessoren- und Speichertypen, die bei einem Einplatinen-Computer gebraucht werden. Er behandelt selbstverständlich nur Typen, die auf dem Markt sind und wohl auch länger bleiben werden. Damit bekommt der Leser zusätzlich eine Anleitung in die Hand, wie er seine Probleme mit einem Einplatinen-Computer lösen kann. Er lernt auch Typisches selbst zu programmieren.

Preisänderungen und Liefermöglichkeit vorbehalten.

Franzis-Verlag, München

Weitere Franzis-Computer-Fachbücher

Z-80-Applikationen

Eine Sammlung von gut dokumentierten Programmen, die universell anwendbar sind. Von Michael **Klein.**
144 Seiten, 89 Abbildungen.
Lwstr-geb. DM 38.–
ISBN 3-7723-6672-4

Das Applikationsbuch ist ein Schritt in die Richtung „Lösen von Standardproblemen".
Bewußt legt der Autor den Schwerpunkt auf die sofortige Anwendbarkeit der dargestellten Methoden und Programme. Er bietet Standardlösungen z. B. für die Ein-/Ausgabe über eine serielle Schnittstelle oder Interruptschaltung, führt Programme zur Meßwertverarbeitung an und zeigt, wie man sich nützliche Arbeitshilfen schaffen kann.

Mikrocomputersysteme

Selbstbau – Programmierung – Anwendung. Von Rolf-Dieter **Klein.**
3., verbesserte Auflage, 159 Seiten, 134 Abbildungen und 12 Tabellen.
Lwstr-geb. DM 36.–
ISBN 3-7723-6383-0

Kaum zu glauben, daß ein Mikrocomputer im Selbstbau hergestellt werden kann! Daß dieses Vorhaben glückte, hat der Autor bewiesen. Wie ein hinreichend ausgebildeter Elektroniker das nachvollziehen kann, wird in dem Buch hier dargestellt.
Zunächst wird die Hardware geschaffen. Eingabetastatur, Mikroprozessor, Speicher verschiedener Art, Drucker, Sichtgerät, deren Elemente zu einer funktionierenden Einheit zusammengeschlossen werden. Danach die Software. Da zeigt der Autor mehrere Möglichkeiten auf. Nicht etwa nur ein kleines Programm, das immer wieder stupide abläuft. Nein, ausführliche Programme werden vorgestellt, die Spiele und mathematische Aufgaben lösen können.
Als Abschluß und Höhepunkt fügt der Autor Anregungen hinzu, selbst Programme zu schreiben und in dem eigenen Mikrocomputer zu erproben. Was will man mehr?

Basic für Mikrocomputer

Geräte – Begriffe – Befehle – Programme. Von Herwig **Feichtinger.**
2., neu bearbeitete Auflage. 264 Seiten, 42 Abbildungen.
Lwstr-kart. DM 28.–
ISBN 3-7723-6822-0

Dieses praxisorientierte Buch ist Einführung und Nachschlagewerk zugleich. Begriffe aus der Computer-Fachsprache wie ASCII, RS-232-Schnittstelle oder IEC-Bus werden ebenso ausführlich erläutert wie alle derzeit üblichen Befehlsworte. Marktübliche Basic-Rechner werden einander gegenübergestellt, um vor dem Kauf die Wahl zu erleichtern und um das Anpassen von Programmen an den eigenen Rechner zu ermöglichen. Schließlich findet der Leser handfeste Tips für das Erstellen eigener Programme und Beispiele fertiger Problemlösungen für typische Anwendungsfälle.

IEC-Bus

Die Funktionsweise des IEC-Bus und seine Anwendung in Geräten und Systemen. Von Dr. Anton **Piotrowski.**
300 Seiten, 125 Abbildungen und 95 Tabellen.
Lwstr-geb. DM 48.–
ISBN 3-7723-6951-0

Hier liegt eine fachgerechte Darstellung der Funktionsweise des IEC-Bus und dessen Anwendung in den Geräten und den Systemen vor. Diese gliedert sich in drei Teile: 1. Theoretische Grundlagen der Funktion des IEC-Bus mit ausführlicher Beschreibung der Nachrichten- und der Zustandsdiagramme. 2. Beschreibung der IEC-Interface-Bausteine und deren Anwendungen bei der Realisierung von IEC-Schnittstellen mit Mikroprozessoren. 3. Gerätetechnik, Aufbau und Programmierung von IEC-Bus-Systemen, inklusiv der Fehleranalyse.
Ausgewählte und in sich abgeschlossene praktische Anwendungsbeispiele in Hard- und Software erleichtern dem Techniker und Informatiker die tägliche Arbeit.

Mikrocomputer von A bis Z

Bits und Bytes und andere EDV-Begriffe verständlich gemacht. Von Herwig **Feichtinger.**
176 Seiten, 34 Abbildungen.
Lwstr-geb. DM 24.–
ISBN 3-7723-7061-6

Dieses Begriffswörterbuch dient der täglichen Praxis. Es macht das unumgängliche „Fach-Chinesisch" der Computer-Branche verständlich.
Dieses Begriffswörterbuch führt zu objektiven und sachgerechten Beurteilungen der Herstellerangaben. Das spart unter Umständen die Ausgabe von einigen tausend Mark, weil es zu der richtigen Auswahl des richtigen Computers führt.
Dieses Begriffswörterbuch ist auch ein einfaches, elementares Lehrbuch der Mikrocomputertechnik. Wer von ihr angehaucht worden ist, liest die Definitionen hintereinander wie ein gut gemachtes Fachbuch und hat erheblichen Nutzen davon.

Pascal: Einführung – Programmentwicklung – Strukturen

Ein Arbeitsbuch mit zahlreichen Programmen, Übungen und Aufgaben. Von Jürgen **Plate** und Paul **Wittstock.**
387 Seiten mit 178 Abbildungen.
Lwstr-geb. DM 48.–
ISBN 3-7723-6901-4

Schritt für Schritt wird der Leser in das Programmieren mit Pascal eingeführt. Die Autoren haben sich in die Ahnungslosigkeit des Anfängers hineinversetzt. Sie bringen ihm das besondere Denken des routinierten Programmierers bei. Das Verblüffende dabei ist, sie kommen mit einer einfachen klaren Sprache aus, verabscheuen das EDV-Chinesisch, setzen nichts voraus. Wer sich an dieses Buch heranmacht, meint, es gäbe nichts Einfacheres als Pascal.

Preisänderungen und Liefermöglichkeit vorbehalten.

Franzis-Verlag, München

Busch
Basic für Aufsteiger

Rudolf Busch

Basic für Aufsteiger – das ist das Basic, das nicht mehr in den Handbüchern steht. Wer dieses Buch intus hat, der besitzt einen anspruchsvollen Basic-Wortschatz, der sich sehen lassen kann.

Die Schwerpunkte liegen in der Aufbereitung aller mathematischen Funktionen, die Basic bietet. Es folgen jene, mit denen Strings unter allen nur denkbaren Gesichtspunkten bearbeitet werden können. Das Buch schließt ab mit einer Fülle von Programmen aus Mathematik, Geometrie, Betriebswirtschaft und Datenverarbeitung.

Der Autor greift seine Beispiele aus dem täglichen Leben, kleidet sie oft in lustige Geschichten, denn so lernt der Leser am besten, was er tut und warum er es tut.

ISBN 3-7723-7281-3

Franzis'